# 3D OBJECT MODELING AND RECOGNITION IN PHOTOGRAPHS AND VIDEO

Fredrick H. Rothganger, Ph.D.
Computer Science
University of Illinois at Urbana-Champaign, 2004
Jean Ponce, Adviser

This thesis introduces a novel representation for three-dimensional (3D) objects in terms of local affine-invariant descriptors of their appearance and the spatial relationships between the corresponding *affine regions*. Geometric constraints associated with different views of the same surface patches are combined with a normalized representation of their appearance to guide matching and reconstruction, allowing the acquisition of true 3D models from multiple unregistered images, as well as their recognition in photographs and image sequences. The proposed approach is applied to two domains: 1) Photographs – Models of rigid objects are constructed from photos and recognized in highly cluttered shots taken from arbitrary viewpoints. 2) Video – Dynamic scenes containing multiple moving objects observed by a moving camera are segmented into rigid components, and the 3D models constructed from these components are matched across different image sequences, with application to shot matching.

# 3D OBJECT MODELING AND RECOGNITION IN PHOTOGRAPHS AND VIDEO

BY

FREDRICK H. ROTHGANGER

B.A., Central Bible College, 1990
M.S., University of Massachusetts, Boston, 1997

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

# Abstract

This thesis introduces a novel representation for three-dimensional (3D) objects in terms of local affine-invariant descriptors of their appearance and the spatial relationships between the corresponding *affine regions*. Geometric constraints associated with different views of the same surface patches are combined with a normalized representation of their appearance to guide matching and reconstruction, allowing the acquisition of true 3D models from multiple unregistered images, as well as their recognition in photographs and image sequences. The proposed approach is applied to two domains: 1) Photographs – Models of rigid objects are constructed from photos and recognized in highly cluttered shots taken from arbitrary viewpoints. 2) Video – Dynamic scenes containing multiple moving objects observed by a moving camera are segmented into rigid components, and the 3D models constructed from these components are matched across different image sequences, with application to shot matching.

To the bear
To Justin

# Acknowledgments

Thanks to Svetlana Lazebnik, Jean Ponce, and Cordelia Schmid for discussions that helped develop the key ideas in this thesis. Jeff Erickson shared his insights on interval graphs. Thanks to Martial Hebert and Yann LeCun for helpful discussions, and to David Lowe for his insights on the behavior of the SIFT descriptor.

Thanks also to those who provided data. Josef Sivic provided helpful information on the shot segmentation for "Groundhog Day". Rémi Ronfard and Christine Dratva respectively provided the shot segmentation and helped select shots from "Run Lola Run". Thanks to Kenton McHenry and Kevin Squire for help with photography sessions in the lab.

Thanks to Pierre Moreels, Shyjan Mahamud, David Lowe, Mario Munich, and Vittorio Ferrari for testing their recognition systems on the data set presented in Chapter 3, and for providing their data sets to us. Thanks to Akash Kushal for testing our recognition system on the provided data sets.

This research was partially supported by the UIUC Campus Research Board, by the National Science Foundation under grants IRI 99-0709, IIS 03-12438, and IIS 03-08087, by the CNRS-UIUC Research Collaboration Agreements, by the European FET-open project VIBES, and by the UIUC-Toyota collaboration on 3D object modeling, recognition and classification from photographs

# Table of Contents

**Chapter 1 Introduction . . . . . 1**

**Chapter 2 Approach . . . . . 3**
2.1 Affine Regions . . . . . 3
2.1.1 Background . . . . . 4
2.1.2 Detection . . . . . 6
2.1.3 Description . . . . . 7
2.2 Geometric Constraints . . . . . 9
2.2.1 Geometric Interpretation of the Rectification Process . . . . . 9
2.2.2 Affine Multi-view Geometry . . . . . 10
2.2.3 Matching Constraints . . . . . 13
2.2.4 Locally-Affine Projection . . . . . 14
2.3 Matching . . . . . 16
2.4 Discussion . . . . . 19

**Chapter 3 Photographs . . . . . 20**
3.1 Related Work . . . . . 20
3.1.1 Local Feature View Clustering for 3D Object Recognition . . . . . 21
3.1.2 Discriminative Distance Measures for Object Detection . . . . . 22
3.1.3 Image Matching Using Affine-Invariant Image Descriptions . . . . 23
3.2 Modeling . . . . . 24
3.2.1 Image Matching . . . . . 26
3.2.2 Constructing an Integrated Model . . . . . 29
3.2.3 Experimental Results . . . . . 37
3.3 Recognition . . . . . 37
3.3.1 Appearance-Based Selection of Potential Matches . . . . . 39
3.3.2 Estimating Geometry . . . . . 41
3.3.3 Geometry-Based Addition of Matches . . . . . 42
3.3.4 Object Detection . . . . . 43
3.3.5 Experimental Results . . . . . 43
3.4 Discussion . . . . . 48

**Chapter 4 Image Sequences . . . . . 51**
4.1 Background . . . . . 51
4.2 Related Work . . . . . 53
4.2.1 Video Analysis and Shot Matching . . . . . 53
4.2.2 Automated Acquisition of 3D Object Models from Image Sequences 54
4.2.3 Affine Motion Segmentation . . . . . 55
4.3 Modeling . . . . . 57
4.3.1 Tracking . . . . . 57
4.3.2 Motion Segmentation . . . . . 59
4.3.3 Handling Missing Data . . . . . 61
4.3.4 Bilinear Merging . . . . . 62
4.3.5 Results . . . . . 64
4.4 Recognition . . . . . 66
4.4.1 Appearance-Based Selection of Potential Matches . . . . . 68
4.4.2 Robust Estimation . . . . . 69
4.4.3 Geometry-Based Addition of Matches . . . . . 69
4.4.4 Object Detection . . . . . 69
4.4.5 Results . . . . . 70
4.5 Discussion . . . . . 70

**Chapter 5 Discussion . . . . . 76**
5.1 Thesis Contributions . . . . . 76
5.2 Future Work . . . . . 77

**Appendix A Inverse Projection Matrices . . . . . 78**

**Appendix B Patch Refinement . . . . . 80**

**Appendix C CD of Video Results . . . . . 82**

**References . . . . . 83**

**Author's Biography . . . . . 95**

# Chapter 1

# Introduction

This thesis addresses the problem of recognizing 3D objects in photographs and image sequences. Traditional feature-based geometric approaches to this problem—such as alignment [5, 33, 49, 56, 69] or geometric hashing [61, 62, 128]—enumerate various subsets of geometric image features before using pose consistency constraints to confirm or discard competing match hypotheses, but they largely ignore the rich source of information contained in the image brightness and/or color pattern, and thus typically lack an effective mechanism for selecting promising matches. Appearance-based methods—as originally proposed in the context of face recognition [8, 97, 133] and 3D object recognition [91, 118]—take the opposite view, and prefer to explicit geometric reasoning a classical pattern recognition framework [30] that exploits the discriminatory power of (relatively) low-dimensional, empirical models of global object appearance in classification tasks. However, they typically deemphasize the combinatorial aspects of the search involved in any matching task, which limits their ability to handle occlusion and clutter.

Viewpoint and/or illumination invariants (or *invariants* for short) provide a natural indexing mechanism for object recognition tasks. Unfortunately, although planar objects and certain simple shapes—such as bilateral symmetries [92] or various types of generalized cylinders [68, 102]—admit invariants, general 3D shapes do not [15], which is the main reason why invariants have fallen out of favor after an intense flurry of activity in the early 1990s [89, 90]. We propose to revisit invariants as a *local* description of truly three-dimensional objects: Indeed, although smooth surfaces are almost never planar in the large, they are are always planar in the small; that is, sufficiently small patches can be treated as being comprised of coplanar points.[1]

*The central goal of this thesis is to establish a new framework for object recognition*

[1] Physical surfaces are not ideal mathematically smooth ones, but we treat them as such, which indicates that we work with them at an appropriate granularity level.

*where object models consist of a collection of (small) planar patches and a description of their 3D spatial relationships, along with a normalized description of their appearance.* Appearance provides an effective filter for selecting promising match candidates in modeling and recognition tasks, and the 3D spatial relationships afford efficient matching algorithms for discarding geometrically inconsistent candidate matches.

We use local image descriptors that are invariant under affine transformations of the spatial domain [7, 44, 66, 84, 113] and of the brightness/color signal [70] to capture the appearance of salient surface patches. We use a set of multi-view geometric constraints related to those studied in the structure from motion literature [129] to capture their spatial relationship. This approach is directly related to a number of recent techniques that combine local models of image appearance in the neighborhood of salient features—or "interest points" [52]—with local and/or global geometric constraints in wide-baseline stereo matching [127, 135], image retrieval [104, 115], and object recognition tasks [34, 70, 76, 143]. These methods normally either require storing a large number of views for each object [70, 76, 104, 115], or limiting the range of admissible viewpoints [34, 117, 143]. In contrast, our approach supports the automatic acquisition of explicit 3D object models from multiple unregistered images, and their recognition in heavily cluttered pictures taken from arbitrary viewpoints.

We apply and validate the proposed approach on two concrete object recognition problems. The first is the automated modeling and recognition of rigid 3D objects in photographs. The second is modeling and matching of rigid components in image sequences that may contain multiple moving objects observed by moving cameras.

The main scientific contributions of this thesis are:

1. A unified framework for 3D object recognition that combines the advantages of geometric and appearance-based approaches to recognition.

2. An algorithm for automatically acquiring 3D models of rigid objects from a small set of unregistered photographs and recognizing them in cluttered photographs taken from unconstrained viewpoints.

3. An algorithm for finding the rigid parts of an image sequence, constructing 3D models of these parts, and matching them across video clips.

This thesis begins by describing our framework for 3D object modeling and recognition, along with background and related work (Chapter 2). It then describes the specific cases of photographs (Chapter 3) and image sequences (Chapter 4). Finally, it gives some general discussion and points to future work (Chapter 5).

# Chapter 2

# Approach

As noted in the previous chapter, the central goal of this thesis is to establish a new framework for object recognition where object models consist of a collection of planar patches arranged in 3D space, along with a normalized description of their appearance. The approach consists of three key components: (1) the affine regions that provide us with a normalized, viewpoint-independent description of local image appearance; (2) the geometric multi-view constraints associated with the corresponding surface patches; and (3) the algorithms that enforce both photometric and geometric consistency constraints while matching groups of patches in modeling and recognition tasks.

This approach is an offspring of recent work on wide-baseline matching, which in turn depends on the detection and descriptions of image patches in a manner that is repeatable under viewpoint and illumination changes. This chapter reviews these methods, and then introduces the new geometric constraints associated with multiple views of affine-invariant patches that will be used repeatedly in this thesis in matching and motion segmentation tasks. Finally, it presents the the matching algorithm used in various forms throughout this thesis.

## 2.1 Affine Regions

The construction of local invariant models of object appearance involves two steps, the *detection* of salient image regions, and their *description*. Ideally, the regions found in two images of the same object should be the projections of the same surface patches. Therefore, they must be *covariant*, with regions detected in the first picture mapping onto those found in the second one via the geometric and photometric transformations induced by the corresponding viewpoint and illumination changes. In turn, detection must be followed by a description stage that constructs a region representation *invariant* under these changes.

For small patches of smooth Lambertian surfaces, the transformations are (to first order) affine, and this section presents the approach to the detection and description of *affine regions* [44, 84] used in our implementation.

### 2.1.1 Background

Local image descriptors map the pixel values within some small image region onto a feature vector. As the viewpoint changes, the appearance of surface patches undergo systematic variations, and much effort in the past two decades has been devoted to the construction of descriptors that yield the same feature vector irrespective of viewing conditions. There has been steady progress in this area, from determining the location of projected points repeatably [52, 116], to handling more and more of the viewing parameters, including scale [66, 71], shape [7, 67, 84, 124] and orientation [113, 115] of the neighborhood around a point.

These approaches remove the effects of viewpoint variation by applying some combination of two distinct processes. The first is a preprocessing step which directly manipulates the pixels of the patch, registering them into a normalized form. The second process is the mapping from pixel values to feature vector. This mapping can treat different variants of a patch as belonging to an equivalence class, and produce a common feature vector regardless of the variant.

"Interest point" operators handle the problem of locating a point on the surface of an object after it has been projected into an image. Desirable characteristics of a point detector are saliency and repeatability across changes of viewpoint. Harris and Stephens [52] proposed a method of finding salient points which turned out to be more repeatable than several other interest point operators [116]. Harris points are essentially local maxima of the product of the eigenvalues of the second moment matrix of the intensity gradient, though in practice the point finder uses an approximation to avoid computing eigenvalues.

Schmid and Mohr [115] developed rotation invariant descriptors based on various combinations of derivatives around the interest point. Koenderink and van Doorn [59] called the set of such Gaussian derivatives at a point the "local jet" (a term they attribute to Poston and Stewart [105]). Generally, the local jet consists of a truncated Taylor expansion of the intensity function in terms of Gaussian derivatives. The Gaussian derivatives themselves are not rotation-invariant, but Schmid and Mohr showed how to combine them to produce rotation-invariant values.

Scale-space theory led to the development of scale-invariant interest points [66, 71]. These interest points are scale invariant in two senses. First, the location is found at an

appropriate scale level rather than a single fixed scale. Second, the point has a scale attribute as well as a location. Mikolajczyk and Schmid [83] introduced a combined Harris-Laplacian detector that finds Harris points in scale-space but chooses the characteristic scale based on the response of the normalized Laplacian. Combined with some rotation-invariant descriptor, such as the one proposed by Schmid and Mohr, these points achieve two of the requisite types of invariance.

The technique of affine adaptation grew out of a method to find a planar scene patch such that the back-projection of texture from the image onto that scene patch is isotropic [67, 124], in the sense that the intensity gradient has equal variance in all directions. Lindeberg and Gårding [67] proposed instead to deform the image texture directly to make it isotropic. This approach removes variation due to non-uniform scaling and skew by transforming the shape of the patch. Lindeberg proposed an iterative process which alternates between estimating the second moment matrix on the adapted patch and updating the adapting transformation.

Baumberg [7] applied the technique of affine adaptation to build fully affine-invariant descriptors. The affine-adapted interest points provided the invariance to scale, non-uniform scaling and skew, while the descriptor itself provided the rotation invariance. Mikolajczyk and Schmid [84] carried affine adaptation a step further by allowing the scale and location of the interest point to change during the iterative process, based on the observation that both are affected by the deformation of the texture determined by the iterative process.

**Alternatives to Affine Adaptation**

Tuytelaars and Van Gool [135, 136] proposed two alternative approach to finding affine-covariant regions. One is based on forming parallelograms from three points in a repeatable manner. They first anchor one vertex at an interest point and then follow the two strongest edges in the neighborhood to locate the other two vertices. To determine the final positions of the two vertices, and thus the size and shape of the region, they search for the extrema of certain functions (moments) on the texture inside the delineated region. The advantage of this method is that such parallelograms tend not to cross the boundaries of the object. The second method finds elliptical regions around interest points in a repeatable manner. The method involves finding an extremum of a function on the one-dimensional texture along a line through the interest point. The extremum defines a point along that line. After accumulating the points for a number of such lines, they estimate the ellipse that best fits all of them.

Matas et al. [81] proposed finding regions in the image based on intensity thresholding.

Consider an image of intensity values that is binarized at a certain threshold level. As the threshold varies, the boundary between the black and white regions shifts. The shape of the boundary around a given contiguous region (whether black or white) is a function of the threshold. Therefore, the area of that region is also a function of the threshold. A Maximally Stable Extremal Region (MSER) is a contiguous region found at a threshold setting such that the rate of change in its area with respect to the threshold is at a minimum. That is, the shape of the region changes relatively little over a wide range of threshold values.

Tell and Carlson [127] describe a one-dimensional set of pixels rather than a patch. Specifically, they compute a vector of Fourier coefficients from the pixels along a line segment between two interest points. All lines are parameterized so that the Fourier transform is independent of their length. Provided both points are projected from a planar surface in the scene, this description is fully affine-invariant.

### 2.1.2 Detection

This thesis uses a form (Algorithm 1) of the affine-covariant region detector developed by Mikolajczyk and Schmid [84]. This algorithm depends on a separate interest point detector to provide a set of points along with their initial scales. A study by Mikolajczyk et al. [82] concludes that no single detector outperforms the others on all types of scenes and image transformations. Therefore, in the absence of prior knowledge about the type of scene, it is beneficial to use a battery of complementary detectors. The primary detectors we use are the Harris-Laplacian detector and the difference-of-Gaussians (DoG) operator [25, 70, 142]. The Harris detector tends to find corners and points at which significant intensity changes occur (considered to be regions of "high information content" [84]) while the DoG detector is in general attracted to the centers of roughly uniform regions (blobs). Figure 2.1 shows examples of the outputs of these two detectors.

Our implementation of affine adaptation makes two modifications to the one proposed by Mikolajczyk and Schmid. First, we update the location of blob-like regions using the Laplacian detector rather than the Harris detector. Second, we compute an orientation for each patch. The standard output of affine adaptation are elliptical-shaped patches. It is easy to show that any ellipse can be mapped onto a unit circle centered at the origin using a one-parameter family of affine transformations separated from each other by arbitrary orthogonal transformations (intuitively, this follows from the fact that circles are unchanged by rotations and reflections about their centers). This ambiguity can be resolved by determining the dominant gradient orientation of the image region, turning the corresponding

**Input:** The image $L$ and a point $\boldsymbol{x}$ in $L$.
**Output:** A $3 \times 3$ matrix $\mathcal{R}$ that transforms the patch around $\boldsymbol{x}$ into a normalized form.

Initialize a $2 \times 2$ matrix $\mathcal{U}$ to the identity. $\mathcal{U}$ maps coordinates in the original image $L$ into coordinates in a transformed image $L'$. Perform all subsequent steps on the neighborhood around $\boldsymbol{x}$ transformed by the current value of $\mathcal{U}$.

**repeat**

- Determine the characteristic scale $s$ of $\boldsymbol{x}$ by finding the scale of normalized Laplacian with strongest response at $\boldsymbol{x}$.
- Update $\boldsymbol{x}$ by finding the nearest Harris (respectively Laplacian) point within the patch. (If $s$ and $\mathcal{U}$ did not change, the nearest point would be exactly $\boldsymbol{x}$.)
- Estimate the second-moment matrix $\mu$ in the neighborhood of $\boldsymbol{x}$.
- Update $\mathcal{U}$ to make the current neighborhood isotropic: $\mathcal{U} \leftarrow \mu^{1/2}\mathcal{U}$.
- Normalize the determinant of $\mathcal{U}$ to 1.

**until** Very little change in $\mathcal{U}$.

Determine the orientation $\theta$ of the image gradient.

Determine $\mathcal{R}$ by combining all the transformations:

$$\mathcal{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{s}\mathcal{U} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathcal{I} & -\boldsymbol{x} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

**Algorithm 1:** Affine Adaptation.

ellipse into a parallelogram and the unit circle into a square (Figure 2.2). Thus, the output of the detection process is a set of image regions in the shape of parallelograms, described by affine *rectifying transformations* that map each parallelogram onto a "unit" square centered at the origin (Figure 2.3).

### 2.1.3 Description

A rectified affine region is a *normalized* representation of the local surface appearance, invariant under planar affine transformations (Figure 2.4). Under affine—that is, orthographic, weak-perspective, or para-perspective—projection models, this representation is invariant under arbitrary viewpoint changes. For Lambertian patches and distant light sources, it can also be made invariant to changes in illumination (ignoring shadows) by subtracting the mean patch intensity from each pixel value and normalizing the Frobenius norm of the corresponding image array to one. Equivalently, normalized correlation can be used to compare rectified patches, irrespective of viewpoint and (affine) illumination changes. Maximizing correlation is equivalent to minimizing the squared distance between

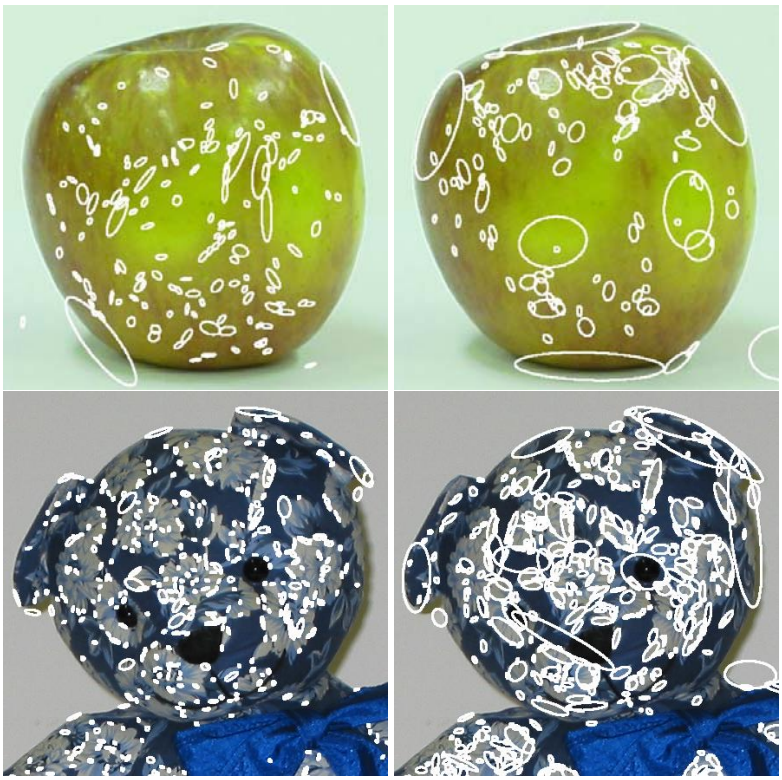


Figure 2.1: Affine-adapted patches found by Harris-Laplacian (left) and DoG (right) detectors.

feature vectors formed by mapping every pixel value onto a separate vector coordinate. Other feature spaces may of course be used as well. In particular, the SIFT descriptor introduced by Lowe [70] has been shown to provide superior performance in image retrieval tasks [85]. Briefly, the SIFT description of an image region is a three-dimensional histogram over the spatial image dimensions and the gradient orientations, with the original rectangular area broken into 16 smaller ones, and the gradient directions quantized into 8 bins (Figure 2.5), and it can thus be represented by a 128-dimensional feature vector [70].

In practice, our experiments have shown that combining the SIFT descriptor with a color histogram improves the recognition rate in difficult cases with low-contrast patches. We build color histograms using a color space in which intensity is truly orthogonal to chroma, specifically YUV space although other possibilities (e.g., XYZ) exist. The histogram is two-dimensional (typically $10 \times 10$) and built only from the chroma component, that is, the U and V values. See Figures 2.5 and 3.12 for examples of the color histograms.

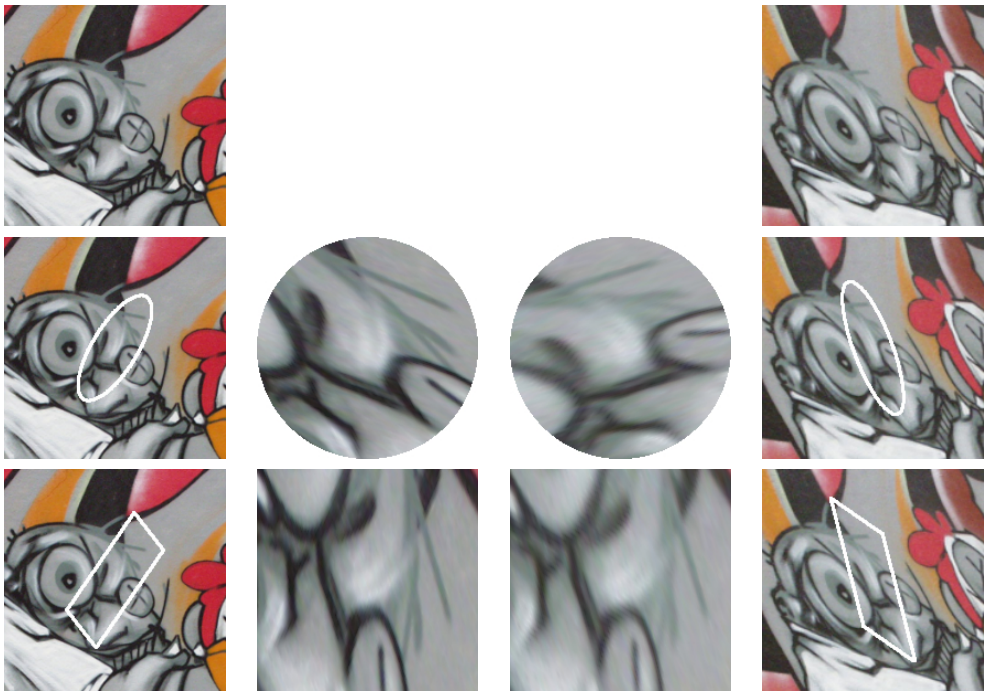


Figure 2.2: Normalizing patches. The left two columns show a patch from image 1 of Krystian Mikolajczyk's graffiti dataset. The right two columns show the matching patch from image 4. The first row shows the region of the original image. The second row shows the ellipse determined by affine adaptation. This normalizes the shape, but leaves a rotation ambiguity, as illustrated by the normalized circles in the center. The last row shows the same patches with orientation determined by the gradient at about twice the characteristic scale.

## 2.2 Geometric Constraints

### 2.2.1 Geometric Interpretation of the Rectification Process

Let us denote by $\mathcal{R}$ and $\mathcal{S} = \mathcal{R}^{-1}$ the rectifying transformation associated with an affine region and its inverse. The matrix $\mathcal{S}$ enjoys a simple geometric interpretation, illustrated by Figure 2.3 (bottom right), that will prove extremely useful in the sequel. Specifically, the form of $\mathcal{S}$ is

$$\mathcal{S} = \begin{bmatrix} \boldsymbol{h} & \boldsymbol{v} & \boldsymbol{c} \\ 0 & 0 & 1 \end{bmatrix}.$$

The matrix $\mathcal{R}$ is an affine transformation from the image patch to its rectified form, and thus $\mathcal{S}$ is an affine transformation from the rectified form back to the image patch. Examining key points in the rectified patch indicates the interpretation of the columns of $\mathcal{S}$. The center of the rectified patch is $[0, 0, 1]^T$. Therefore, the third column of $\mathcal{S}$ gives the homogeneous coordinates of the patch center in the image. The point where the positive $x$-axis pierces the

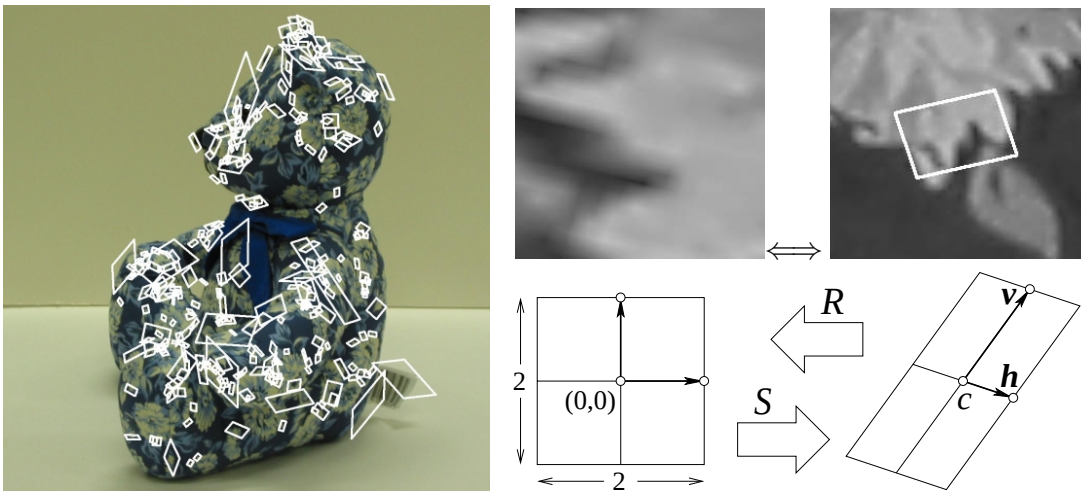



Figure 2.3: Affine regions. Left: A sample of the regions found in an image of a teddy bear (most of the patches actually detected in this image are omitted for clarity). Top right: A rectified patch and the original image region. Bottom right: Geometric interpretation of the rectification matrix $\mathcal{R}$ and its inverse $\mathcal{S}$ (see Section 2.2 for details).

side of the rectified patch is $[1, 0, 1]^T$, and similarly $[0, 1, 1]^T$ for the $y$-axis. In the image, these points are respectively $\begin{bmatrix} \boldsymbol{h} + \boldsymbol{c} \\ 1 \end{bmatrix}$ and $\begin{bmatrix} \boldsymbol{v} + \boldsymbol{c} \\ 1 \end{bmatrix}$, and it is easy to see that $\boldsymbol{h}$ and $\boldsymbol{v}$ are vectors joining $\begin{bmatrix} \boldsymbol{c} \\ 1 \end{bmatrix}$ to the sides of the corresponding parallelogram (Figure 2.3).

The matrix $\mathcal{S}$ effectively contains the locations of three points in the image, so a match between $m \geq 2$ images of the same patch contains *exactly* the same information as a match between $m$ triples of points. It is thus clear that all the machinery of structure from motion [129] and pose estimation [56, 69] from point matches can be exploited in modeling and object recognition tasks. Reasoning in terms of multi-view constraints associated with the matrix $\mathcal{S}$ will provide in the next section a unified and convenient representation for all stages of both tasks, but one should always keep in mind the simple geometric interpretation of the matrix $\mathcal{S}$ and the deeply rooted relationship between these constraints and those used in motion analysis and pose estimation.

### 2.2.2 Affine Multi-view Geometry

Let us assume for the time being that we are given $n$ patches observed in $m$ images, together with the (inverse) rectifying transformations $\mathcal{S}_{ij}$ defined as in the previous section for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ ($i$ and $j$ serving respectively as image and patch indices). We use these matrices to derive in this section a set of geometric and algebraic constraints that

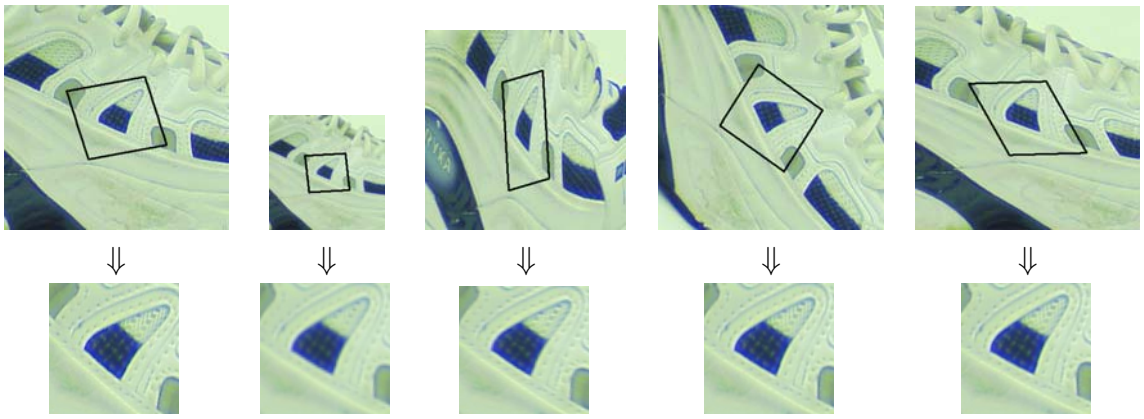


Figure 2.4: Rectifying various deformations. Top: the patch in the context of a deformed image. Bottom: the rectified form of the patch. Left to right: original image, uniform scaling, non-uniform scaling, rotation, skew.

must be satisfied by matching image regions.

A rectified patch can be thought of as another view of the original surface patch (Figure 2.6), and the mapping $\mathcal{S}_{ij}$ can thus be decomposed into an *inverse projection* $\mathcal{N}_j$ [32] that maps the rectified patch onto the corresponding surface patch, followed by a projection $\mathcal{M}_i$ that maps that patch onto its projection in image number $i$. In particular, we can write $\mathcal{S}_{ij} = \mathcal{M}_i\mathcal{N}_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, or, in a more compact form:

$$\hat{\mathcal{S}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{S}_{11} & \ldots & \mathcal{S}_{1n} \\ \vdots & \ddots & \vdots \\ \mathcal{S}_{m1} & \ldots & \mathcal{S}_{mn} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_1 \\ \vdots \\ \mathcal{M}_m \end{bmatrix} \begin{bmatrix} \mathcal{N}_1 & \ldots & \mathcal{N}_n \end{bmatrix},$$

and it follows that the $3m \times 3n$ matrix $\hat{\mathcal{S}}$ has at most rank 4.

As shown in Appendix A, the inverse projection matrix can be written as

$$\mathcal{N}_j = \begin{bmatrix} \boldsymbol{H}_j & \boldsymbol{V}_j & \boldsymbol{C}_j \\ 0 & 0 & 1 \end{bmatrix},$$

and it satisfies the constraint $\mathcal{N}_j^T\boldsymbol{\Pi}_j = \mathbf{0}$, where $\boldsymbol{\Pi}_j$ is the coordinate vector of the plane $\Pi_j$ that contains the patch. In addition, the columns of the matrix $\mathcal{N}_j$ admit in our case a geometric interpretation related to that of the matrix $\mathcal{S}_{ij}$: Namely, the first two contain the "horizontal" and "vertical" axes of the surface patch, and the third one is the homogeneous coordinate vector of its center.

To account for the form of $\mathcal{N}_j$, we construct a reduced factorization of $\hat{\mathcal{S}}$ by picking, as in [129], the center of mass of the observed patches' centers as the origin of the world

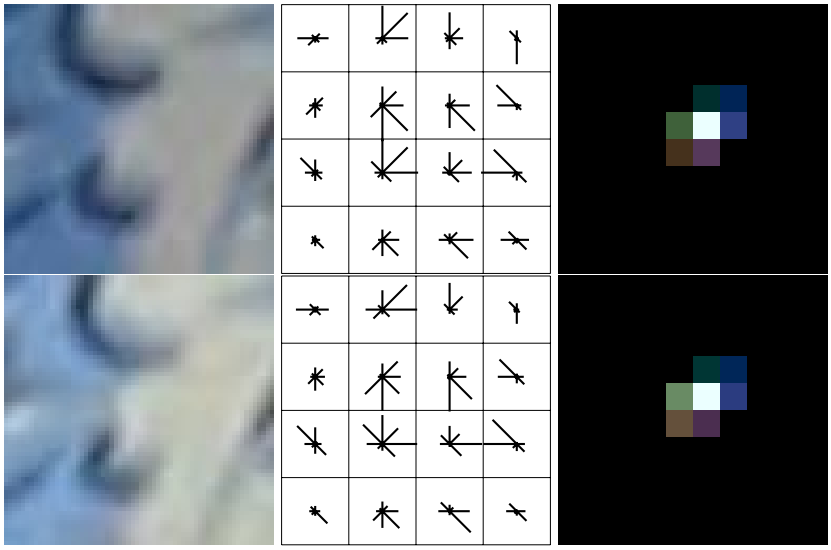


Figure 2.5: Two (rectified) matching patches found in two images of a teddy bear, along with the corresponding SIFT and color descriptors. Here (as in Figure 3.12 later), the orientation histogram values associated with each spatial bin are depicted by lines of different lengths for each one of the 8 quantized gradient orientations. As recommended in [70], we scale the feature vectors associated with SIFT descriptors to unit norm, and compare them using the Euclidean distance. In this example, the distance is 0.28. The (monochrome) correlation of the two rectified patches is 0.9, and the $\chi^2$ distance between the color histograms (explained in Section 3.3.1) is 0.28.

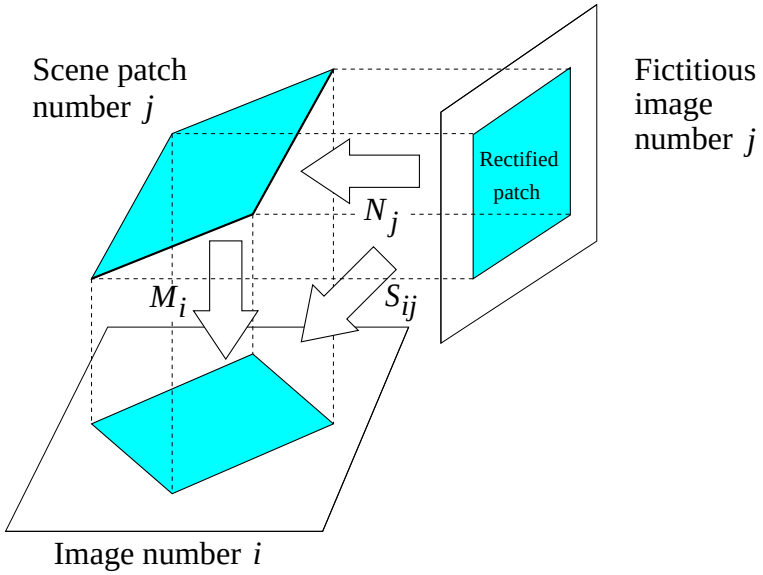



Figure 2.6: Geometric interpretation of the decomposition of the mapping $\mathcal{S}_{ij}$ into the product of a projection matrix $\mathcal{M}_i$ and an inverse projection matrix $\mathcal{N}_j$.

coordinate system, and the center of mass of these points' projections as the origin of every image coordinate system. In this case, the projection equation $\mathcal{S}_{ij} = \mathcal{M}_i \mathcal{N}_j$ becomes

$$\begin{bmatrix} \mathcal{D}_{ij} \\ 0\ 0\ 1 \end{bmatrix} = \begin{bmatrix} \mathcal{A}_i & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathcal{B}_j \\ 0\ 0\ 1 \end{bmatrix}, \quad \text{or} \quad \mathcal{D}_{ij} = \mathcal{A}_i \mathcal{B}_j,$$

where $\mathcal{A}_i$ is a $2 \times 3$ matrix, $\mathcal{D}_{ij} = [\boldsymbol{h}_{ij}\ \boldsymbol{v}_{ij}\ \boldsymbol{c}_{ij}]$ is a $2 \times 3$ matrix, and $\mathcal{B}_j = [\boldsymbol{H}_j\ \boldsymbol{V}_j\ \boldsymbol{C}_j]$ is a $3 \times 3$ matrix. It follows that the reduced $2m \times 3n$ matrix

$$\hat{\mathcal{D}} = \hat{\mathcal{A}}\hat{\mathcal{B}}, \text{ where } \hat{\mathcal{D}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{D}_{11} & \dots & \mathcal{D}_{1n} \\ \vdots & \ddots & \vdots \\ \mathcal{D}_{m1} & \dots & \mathcal{D}_{mn} \end{bmatrix}, \ \hat{\mathcal{A}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{A}_1 \\ \vdots \\ \mathcal{A}_m \end{bmatrix}, \ \hat{\mathcal{B}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathcal{B}_1 & \dots & \mathcal{B}_n \end{bmatrix}, \quad (2.1)$$

has at most rank 3.

### 2.2.3 Matching Constraints

The rank deficiency of the matrix $\hat{\mathcal{D}}$ can be used as a geometric consistency constraint when at least two potential matches are visible in at least two views. Alternatively, singular value decomposition can be used, as in [129], to factorize $\hat{\mathcal{D}}$ and compute estimates of the matrices $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ that minimize the squared Frobenius norm of the matrix $\hat{\mathcal{D}} - \hat{\mathcal{A}}\hat{\mathcal{B}}$. Geometrically, the (normalized) Frobenius norm $d = |\hat{\mathcal{D}} - \hat{\mathcal{A}}\hat{\mathcal{B}}|/\sqrt{3mn}$ of the residual can be interpreted as the root-mean-squared distance (in pixels) between the center and normalized side points of the patches observed in the image and those predicted from the recovered matrices $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$. Given $n$ matches established across $m$ images (a match is an $m$-tuple of image patches), the residual error $d$ can thus be used as a measure of *inconsistency* between the matches.

Together with the normalized models of local shape and appearance proposed in Section 2.1.3, this measure will prove an essential ingredient of the approach to (pairwise) image matching presented in the next chapter. It will also prove useful in modeling tasks where the projection matrices are known but the 3D configuration $\mathcal{B}$ of a single patch is unknown, and in recognition tasks when the patches' configurations are known but a single projection matrix $\mathcal{A}$ is unknown. In general, Eq. (2.1) provides an over-constrained set of linear equations on the unknown parameters of the matrix $\mathcal{B}$ in the former case, and an over-constrained set of linear constraints on the unknown parameters of the matrix $\mathcal{A}$ in the latter one. Both are easily solved using linear least-squares, and they determine the corresponding value of the residual error.

### 2.2.4 Locally-Affine Projection

It is in fact also possible to mix local affine constraints with global perspective ones: Indeed, for patches whose relief is small compared to the distance separating them from the camera, the local projective distortions associated with the perspective projection process are normally negligible, and the rectifying transformations can thus be modeled as planar homographies that just happen to have an affine form (see [135] for related work in the image matching domain). It is easy to show that this amounts to using a variant of weak-perspective or para-perspective projection where the reference depth $z_{ij}$ varies from patch to patch.

One approach to obtaining a locally-affine model is to linearize the perspective projection equation in the neighborhood of the patch center. Consider the homogeneous projection equation

$$\begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} = \frac{1}{z}\mathcal{M}\begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix}, \quad \text{where} \quad \mathcal{M} = \begin{bmatrix} \mathcal{A} & \boldsymbol{b} \\ \boldsymbol{a}_3^T & 1 \end{bmatrix}$$

is the perspective projection matrix, $\mathcal{A}$ is a $2\times3$ sub-matrix of $\mathcal{M}$, $\boldsymbol{p}$ is the non-homogeneous coordinate vector for the point in the image, and $\boldsymbol{P}$ is the non-homogeneous coordinate vector of the point in 3D. We can write the perspective projection mapping as

$$\boldsymbol{p} = f(\boldsymbol{P}) = \frac{1}{\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1}(\mathcal{A}\boldsymbol{P} + \boldsymbol{b}),$$

and a Taylor expansion of order 1 of the function $f$ in $\boldsymbol{P}$ yields $f(\boldsymbol{P} + \boldsymbol{\delta P}) = \boldsymbol{p} + \boldsymbol{\delta p} = f(\boldsymbol{P}) + f'(\boldsymbol{P})\boldsymbol{\delta P}$, or

$$\begin{aligned} \boldsymbol{\delta p} &= f'(\boldsymbol{P})\boldsymbol{\delta P} \\ &= \frac{\mathcal{A}(\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1) - (\mathcal{A}\boldsymbol{P} + \boldsymbol{b})\boldsymbol{a}_3^T}{(\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1)^2}\boldsymbol{\delta P} \\ &= \frac{1}{\boldsymbol{a}_3 \cdot \boldsymbol{P} + 1}(\mathcal{A} - \boldsymbol{p}\boldsymbol{a}_3^T)\boldsymbol{\delta P}. \end{aligned} \tag{2.2}$$

The basis vectors $\boldsymbol{H}$ and $\boldsymbol{V}$ of the 3D patch are essentially small changes around the patch center $\boldsymbol{C}$, so they play the role of $\boldsymbol{\delta P}$. The projection of a 3D patch into an image is then

$$\begin{cases} \boldsymbol{h} &= f'(\boldsymbol{C})\boldsymbol{H}, \\ \boldsymbol{v} &= f'(\boldsymbol{C})\boldsymbol{V}, \\ \boldsymbol{c} &= f(\boldsymbol{C}). \end{cases} \tag{2.3}$$

Since these equations are non-linear, direct factorization is not applicable. We form initial estimates of the cameras and patches using the affine setup described in Section 2.2.2, but then use the iterative process given by Algorithm 2 to search for a set of camera and patch values which minimize the reprojection error under the locally-affine projection model. This algorithm works by holding one set of parameters fixed while estimating the others using linear least squares. By alternating sets of parameters, it is able to update the estimates for all of them once per iteration and eventually converge to a local minimum [78, 132]. Note that, unlike factorization, this method is readily adapted to the case where some patches are only visible in some of the images.

**Input:**
• Image measurements $\mathcal{S}_{ij}$ ($i = 1, \ldots, m$ and $j = 1, \ldots, n$), possibly sparse.
• Appropriate definitions for the camera equations and patch equations.
**Output:** Camera matrices $\mathcal{M}_i$ and patch matrices $\mathcal{B}_j$.

```
Initialize the vectors B_j for all j using the affine method described in section 2.2.2.
repeat
  for i = 1, ..., m do
    Solve for M_i by stacking the n_i instances of the camera equation associated with
    the patches observed in image i.
  end for
  for j = 1, ..., n do
    Solve for B_j by stacking the m_j instances of the patch equation associated with the
    images containing patch j.
  end for
until convergence
```

**Algorithm 2:** Bilinear Iterations.

Algorithm 2 depends on having a set of linear equations for the cameras in terms of known patches, and a set of linear equations for the patches in terms of known cameras. We derive these by first expanding the Eqs. (2.3) to yield

$$(\boldsymbol{a}_3 \cdot \boldsymbol{C} + 1) \begin{bmatrix} \boldsymbol{h} & \boldsymbol{v} \end{bmatrix} = (\mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T) \begin{bmatrix} \boldsymbol{H} & \boldsymbol{V} \end{bmatrix}, \tag{2.4}$$

and

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{a}_3 \cdot \boldsymbol{C} + 1) &= \mathcal{A}\boldsymbol{C} + \boldsymbol{b}, \quad \text{or:} \\ \boldsymbol{c} &= (\mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T)\boldsymbol{C} + \boldsymbol{b}. \end{aligned} \tag{2.5}$$

Given a fixed projection matrix $\mathcal{M}$, putting Eqs. (2.4) and (2.5) together now yields a

system of 6 linear equations in the 9 unknown coordinates of $\boldsymbol{H}$, $\boldsymbol{V}$, and $\boldsymbol{C}$:

$$\left[\begin{array}{c|c|c} \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T & \mathbf{0}^T & -\boldsymbol{h}\boldsymbol{a}_3^T \\ \mathbf{0}^T & \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T & -\boldsymbol{v}\boldsymbol{a}_3^T \\ \mathbf{0}^T & \mathbf{0}^T & \mathcal{A} - \boldsymbol{c}\boldsymbol{a}_3^T \end{array}\right] \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{V} \\ \boldsymbol{C} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{v} \\ \boldsymbol{c} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{b} \end{bmatrix}. \tag{2.6}$$

Given fixed vectors $\boldsymbol{H}$, $\boldsymbol{V}$, and $\boldsymbol{C}$, Eqs. (2.4) and (2.5) also provide a system of 6 linear equations in the 11 unknown entries of $\mathcal{M}$:

$$\left[\begin{array}{c|c|c} \mathcal{H} & -\boldsymbol{h}\boldsymbol{C}^T - \boldsymbol{c}\boldsymbol{H}^T & 0_2 \\ \mathcal{V} & -\boldsymbol{v}\boldsymbol{C}^T - \boldsymbol{c}\boldsymbol{V}^T & 0_2 \\ \mathcal{C} & -\boldsymbol{c}\boldsymbol{C}^T & \mathcal{I}_2 \end{array}\right] \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{v} \\ \boldsymbol{c} \end{bmatrix}, \tag{2.7}$$

where $0_2$ and $\mathcal{I}_2$ are respectively the $2 \times 2$ zero and identity matrices, $\boldsymbol{a}_1^T$ and $\boldsymbol{a}_2^T$ are the first two rows of $\mathcal{M}_1$, and

$$\mathcal{H} = \begin{bmatrix} \boldsymbol{H}^T & \mathbf{0}^T \\ \mathbf{0}^T & \boldsymbol{H}^T \end{bmatrix}, \; \mathcal{V} = \begin{bmatrix} \boldsymbol{V}^T & \mathbf{0}^T \\ \mathbf{0}^T & \boldsymbol{V}^T \end{bmatrix}, \; \mathcal{C} = \begin{bmatrix} \boldsymbol{C}^T & \mathbf{0}^T \\ \mathbf{0}^T & \boldsymbol{C}^T \end{bmatrix}.$$

Given the ambiguity of projective structure from motion, we have $6mn$ equations in $11m + 9n - 15$ unknowns. These equations are redundant whenever $n \geq 2$ image tracks share at least $m \geq 3$ frames, and it is possible to judge whether the corresponding patches move together rigidly by solving for the structure and motion parameters and measuring as before the mean-squared distance in pixels between the predicted and measured values of the vectors $\boldsymbol{c}_{ij}$, $\boldsymbol{h}_{ij}$, and $\boldsymbol{v}_{ij}$.

## 2.3 Matching

The core computational components of model acquisition and object recognition are matching procedures: we seek matches between two sets of patches that are photometrically and geometrically consistent. Concretely, there are three matching tasks in this thesis:

- Image matching – We seek matches between the affine regions found in two pictures that are consistent with both the local appearance models introduced in Section 2.1.3 and the geometric constraints expressed by Eq. (2.1).

- Object recognition – We seek matches between the 3D patches stored in a model

(in the form of the $\mathcal{N}_j$ matrices discussed in Section 2.2) and the affine regions in a picture. Equation (2.1) again provides the geometric constraints.

- Video shot matching – We seek matches between the 3D patches in two models. Appearance constraints are the same as the above two tasks, and geometric consistency is measured by the distance between matched points in the registered models.

All three tasks can be understood in the *constrained-search* model proposed by Grimson [48], who has shown that finding an optimal solution—maximizing, say, the number of matches such that photometric and geometric discrepancies are bounded by some threshold, or some other reasonable criterion—is in general intractable (i.e., exponential in the number of matched features) in the presence of uncertainty, clutter, and occlusion.

Various approaches to finding a reasonable set of geometrically-consistent matches have been proposed in the past, including *interpretation tree* (or *alignment*) techniques [5, 33, 49, 56, 69], and *geometric hashing* [61, 62]. An alternative is offered by *robust estimation* algorithms, such as *RANSAC* [36], its variants [130], and *median least-squares*, that consider candidate correspondences consistent with a small set of *seed* matches as *inliers* to be retained in a fitting process, while matches exceeding some inconsistency threshold are considered as *outliers* and rejected. Although, like all other heuristic approaches to constrained search, RANSAC and its variants are not guaranteed to output an optimal set of matches, they often offer a good compromise between the number of feature combinations that have to be examined and the pruning capabilities afforded by appearance- and geometry-based constraints: In particular, the number of samples necessary to achieve a desired performance with high probability can easily be computed from estimates of the percentage of inliers in the dataset, and it is independent of the actual size of the dataset [36].

Briefly, RANSAC iterates over two steps: In the *sampling* stage, a (usually, but not always) minimal set of matches is chosen randomly, and this "seed" set is used to estimate the geometric parameters of the fitting problem at hand. The *consensus* stage then adds to the initial seed all the candidate matches that are consistent with the estimated geometry. The process iterates until a sufficiently large consensus set is found, and the geometric parameters are finally re-estimated. Despite the attractive features mentioned in the previous paragraph, pure RANSAC only achieves moderate performance in the challenging object recognition experiments presented in this thesis, where clutter may contribute 90% or more of the detected regions. As will be shown later in the experiments, Algorithm 3 below achieves better results. This algorithm uses the idea of consensus from RANSAC while it seeks the maximal set of consistent matches between two sets of patches. It operates in

three key steps, explained below.

**Input:** Two sets of patches $A$ and $B$.
**Output:** A set $T \subseteq A \times B$ of trusted matches.

*Step 1: Appearance-based selection of potential matches.*
• Initialize the set of matches $M$ by finding patch pairs from $A \times B$ with high appearance similarity.
*Step 2: Robust estimation.*
• Apply robust estimation to find a set $T \subseteq M$ of geometrically consistent ("trusted") matches.
• Use consistency constraints to remove outliers from $T$.
*Step 3: Geometry-based addition of matches.*
**repeat**
  **repeat**
    • Form a geometric model $r$ from $T$.
    • Replace $T$ with all matches in $M$ that are consistent with $r$.
  **until** $T$ stops changing.
  • Use consistency constraints to remove outliers from $T$.
  • Re-estimate $r$ from $T$.
  • Add more putative matches to $M$ using $r$ as a guide.
**until** $M$ stops changing.

**Algorithm 3:** Overall Matching Procedure.

Step 1 of the algorithm takes advantage of appearance constraints to reduce the practical cost of the search. It focuses the matching process on the portion of the space of all matches ($A \times B$) which is *a priori* most likely to be correct. Here we are using appearance similarity as a heuristic, since it cannot be a perfect indicator of correct matches. Noise present in actual image measurements lowers the appearance scores for some true matches. Furthermore, nothing prevents incorrect matches from appearing the same.

Step 2 applies RANSAC to the limited set of match hypotheses to find a geometrically consistent subset. Our assumption is that the largest such consistent set will contain mostly true matches. This establishes the geometric relationship between the two sets of patches. Proceeding to Step 3 is optional but useful, since it enhances the results of the matching process.

Step 3 explores the remainder of the space of all matches, seeking other matches which are consistent with the established geometric relationship between the two sets of patches. Obtaining a (nearly) maximal set of matches is useful for recognition (where the number of matches acts as a confidence measure) and for modeling (where they provide more coverage of the object).

The same overall matching procedure is used in our three matching tasks. Section 3.3 provides an extensive experimental comparison of various alternatives for Step 2, and gives details on our preferred implementation.

## 2.4 Discussion

*The first contribution of this thesis is a framework for recognition built on (small) planar surface patches, their 3D spatial relationships and an invariant description of their appearance.* Affine-covariant patches are image measurements that together with the local planarity property of surface patches provide the means to estimate both an affine-invariant appearance description and the 3D structure of an object. Affine-invariant appearance descriptors filter likely matches between sets of patches. Multi-view constraints measure the consistency of sets of matches. Together, affine-covariant patches and multi-view constraints form a foundation for modeling and recognition in photographs and image sequences. In the case of image sequences, they also provide a means for motion segmentation.

This powerful framework offers a number of syntheses. By directly incorporating the shape of each patch in the multi-view geometric equations (Equations (2.1) and (2.3)), it synthesizes shape-from-texture with structure-from-motion. Furthermore, it synthesizes the single-camera constraint [107] with multi-view geometry by treating pose recovery as another instance of the same multi-view equations.

The approach to detecting patches could be improved by using Maximally Stable Extremal Regions (MSER) rather than (or in addition to) Difference of Gaussian (DoG) regions to complement the Harris detector. Personal communication with Josef Sivic and Vittorio Ferrari indicates that MSER performs well as the "blob" (that is, homogeneous region) detector in a complementary set of detectors.

# Chapter 3

# Photographs

This chapter addresses the problem of modeling and recognizing objects in photographs. Chapter 2 outlined our approach: the detection and description of affine-invariant patches, and the representation of their global arrangement as planar patches in 3D. Here we construct 3D models of objects from sparse collections of photographs and recognize those models in novel images from arbitrary viewpoints. The set of training images for a given object is "sparse" in the sense that the amount of viewpoint change between any pair of images is fairly large, typically greater than 20 degrees, and the total number of training images never exceeds 30. These images do not need to be registered, and are typically uncluttered. The recognition method uses both the appearance of the patches and strong 3D constraints on their shape and arrangement to detect consistent match candidates in a test image. Test images do not need to be registered and may contain clutter and occlusion. We present experimental evaluation of the entire process. Figure 3.1 shows an example with some of the modeled objects and a scene in which they are recognized.

## 3.1 Related Work

Traditional geometric approaches to the recognition of rigid 3D objects from photographs—for example alignment and interpretation trees [49, 56, 69]—enumerate all triples of image features before pose consistency constraints are used to confirm or discard competing match hypotheses. Originally limited to simple shapes such as polyhedra, they have been extended to more general shapes including generalized cylinders [68, 102, 147], algebraic surfaces [58, 60], and even free-form surfaces [57, 119, 125, 140]. Within-class variability has been mostly addressed in the context of structural, part-based object descriptions [10, 14, 41, 42, 79, 94, 121, 147, 150]. Unfortunately, the combinatorial complexity of hypothesis formation [48] (and/or the need for a separate segmentation stage) has limited


Figure 3.1: Results of a recognition experiment. Left: A test image. Right: Instances of five models (a teddy bear, a doll stand, a salt can, a toy truck and a vase) have been recognized, and the models are rendered in the poses estimated by our program. Bounding boxes for the reprojections are shown as black rectangles.

the success of purely geometric recognition techniques in cluttered scenes.

Appearance-based techniques, on the other hand, use rich local descriptions of the image brightness pattern to select a relatively small set of promising potential matches before (if at all) using geometric consistency constraints to retain the correct ones. They do not impose restrictions on the shape of the objects that can be recognized, and they have been applied to scenes that contain complex rigid [17, 77, 91, 115] and articulated [9] 3D objects, as well as instances of object classes such as cars [2, 117, 143], faces [53, 54, 110, 117], and people [96, 108]. Although some approaches require a separate segmentation stage [91, 134], others use a combination of local and semi-local image descriptors to avoid segmentation altogether [9, 71, 115]. By taking advantage of recent advances in machine learning [109, 111, 114, 137], several researchers have obtained robust recognition results in highly-cluttered images [17, 77, 117], and even achieved real-time performance [141]. However, because the systematic variation in appearance due to viewpoint and illumination changes is rarely modeled explicitly, appearance-based approaches to 3D object recognition usually have to use and/or store a large number of training images (e.g., [91, 104, 115, 118]), or to limit the range of admissible viewpoints (e.g., [2, 8, 117, 134, 143]).

We focus in this section on three approaches to image matching and object recognition that are particularly relevant to the work presented in the rest of this chapter.

### 3.1.1 Local Feature View Clustering for 3D Object Recognition

In [72], Lowe models an object as a collection of 2D views. Each view consists of SIFT [71] features and their locations. Features that are similar across views are linked together.

Recognition proceeds in three steps. First, the SIFT features from the input image are matched against the features stored in the various views of the model, and each match votes for a view and pose via a Hough transform [55]. A vote for the closest feature in the model also propagates to the linked features in other views. Second, the locations of the matched features determine a similarity transform between the input image and each view via least squares estimation. Finally, each view receives a probability of correctness based on how many image features appear within the outline of its reprojection and on how likely one such feature is to be mismatched.

The training component of the system depends on the recognition component to identify the closest view currently in the model. As it processes each training view, it takes one of three actions based on whether the view matched a model view and (provided the view did match something) on the goodness of the estimate of the similarity transform: 1) If there is no match between any view of any existing model, the input image forms a new view in a new model. 2) If there is a match, but the estimate of the similarity transform is poor, then the image forms a new view in the existing model. 3) If there is a match with a well-estimated transform to some view, then the image features and the view features are merged, with appropriate updating to the links with other model views.

Moreels et al. [88] propose a similar system to Lowe's, but within a probabilistic framework. They attempt to combine the strengths of the probabilistic constellation model [34, 143] with Lowe's deterministic indexing method. They do not incorporate the idea of storing multiple views of the object connected by associations among the features, but rather attempt to learn the features and probability density functions of a single constellation per object. The key difference with previous constellation approaches is that this one learns some of the parameters of the probability model over all the objects in the database rather than separately for each object. Image processing into SIFT features and matching to features stored in the database proceeds in the same way as Lowe's system. A match hypothesis consists of an assignment of each image feature to some object model feature or to the background, along with estimated poses for the objects. There may be any number of object instances in a given hypothesis. A partial hypothesis in one in which some image features are unassigned. Recognition is an $A^*$ search in a hypothesis tree, where partial hypotheses form internal nodes of a tree, and complete hypotheses are the leaves.

### 3.1.2 Discriminative Distance Measures for Object Detection

Mahamud and Hebert [75, 76] take a part-detection approach to recognition, though in their approach there is no linkage between features in various training views of an object. Their

approach is entirely appearance based, without 3D structure. They store multiple views of an object and verify that the features of a recognized view appear in the image in roughly the same arrangement as in the training view. Unlike [72], they treat each model view independently of the others.

The heart of the approach is a method for doing nearest neighbor (NN) classification on parts (that is, object features) using an optimal distance measure. This measure is optimal in the sense that it attempts to minimize the risk of mis-classification. It is a function of multiple distance measures in simple feature spaces that are combined linearly and then passed through a squashing function. Mahamud and Hebert show that the mis-classification risk as a function of the linear mixing parameters is convex, so they can apply standard numerical methods to find the optimal weights.

### 3.1.3 Image Matching Using Affine-Invariant Image Descriptions

Several recent approaches to image matching use affine-invariant descriptors (discussed in Section 2.1) combined with binocular geometric constraints to recognize objects (modeled directly as a set of stored training images) or to retrieve images from a database.

Tuytelaars and Van Gool [135, 136] find elliptical and rectangular affine-covariant regions regions and compute descriptors of their texture based on moments. They use these features to find matches between the two images, and then apply several constraints to verify the matches. The most interesting constraint is one closely related to the geometric constraint we give in Section 2.2.2. It checks for rigid motion between a pair of matches by testing the rank of a matrix constructed from the homographies induced by the matches.

Tell and Carlsson [127] describe affine-covariant lines (rather than regions) where the endpoints are determined by the Harris detector. They use a voting scheme to determine matching interest points between views. If a line segment in one image matches a line segment in the other, then their endpoints are implicitly matched, and the match at each endpoint receives one vote. Point matches with enough votes become the detected matches. Finally, Tell and Carlsson filter the matches with two constraints. One constraint is a voting scheme that takes five point matches per sample and plugs them into an equation that eliminates all camera parameters (assuming an orthographic camera model). Samples that are consistent according to the equation add a vote to each of the member point matches. The other constraint is RANSAC [36] on affine epipolar geometry estimated from seven point matches.

Schaffalitzky and Zisserman [113] apply wide-baseline matching to the problem of finding the relationships between a set of photographs. They introduce a new local de-

scriptor based on complex moments of Gaussian filter responses. They preprocess a patch to remove all but rotation variance. After computing the descriptor, they remove rotation variance by rotating the patch so the strongest responding moment has a positive real value. The part of their work that is most related to ours is their use of the affine transformation associated with a matched pair of patches to help locate additional matches and to reduce the number of matches needed to estimate the fundamental matrix.

Ferrari et al. [35] also use the affine transformation associated with a matched pair of patches. However, instead of simply searching for nearby interest points in the two images that are likely matches, they generate new interest points in a hexagonal grid pattern around the respective patches of the anchor match. This makes their method less dependent on the repeatability of the region detector. Furthermore, the capacity of an anchor match to generate more consistent matches is itself a measure of the correctness of the anchor match. They make use of this property in an iterative procedure that "explores" the matched area in two views of an object.

Mikolajczyk and Schmid [84] propose a method of affine adaption (used in this thesis) that finds affine-covariant regions up to rotation. They describe the resulting ellipses using a set of normalized Gaussian derivatives, and form putative matches between images based on appearance. RANSAC, combined with a homography or fundamental matrix model, selects a geometrically consistent set of matches. They apply their approach to an image retrieval task.

## 3.2 Modeling

This section presents our approach to the automated acquisition of 3D object models from collections of unregistered photographs. These models consist of collections of 3D surface patches in the shape of parallelograms, along with the appearance of the surface within each patch. We will use the teddy bear shown in Figure 3.2 to illustrate some of the steps of the modeling process. Additional modeling experiments will be presented in Section 3.2.3.

The modeling process starts by establishing matches between patches in nearby pairs of input images. Then it connects these matches together into a global set of matches across all the images. Essentially, this establishes the identity of each patch in all the images where it appears. This provides a sparse data matrix of all patches across all images. The process then constructs models from subsets of the data matrix using one of the methods described in Section 2.2. Finally, it registers these into a global model and refines it with a form of bundle adjustment.

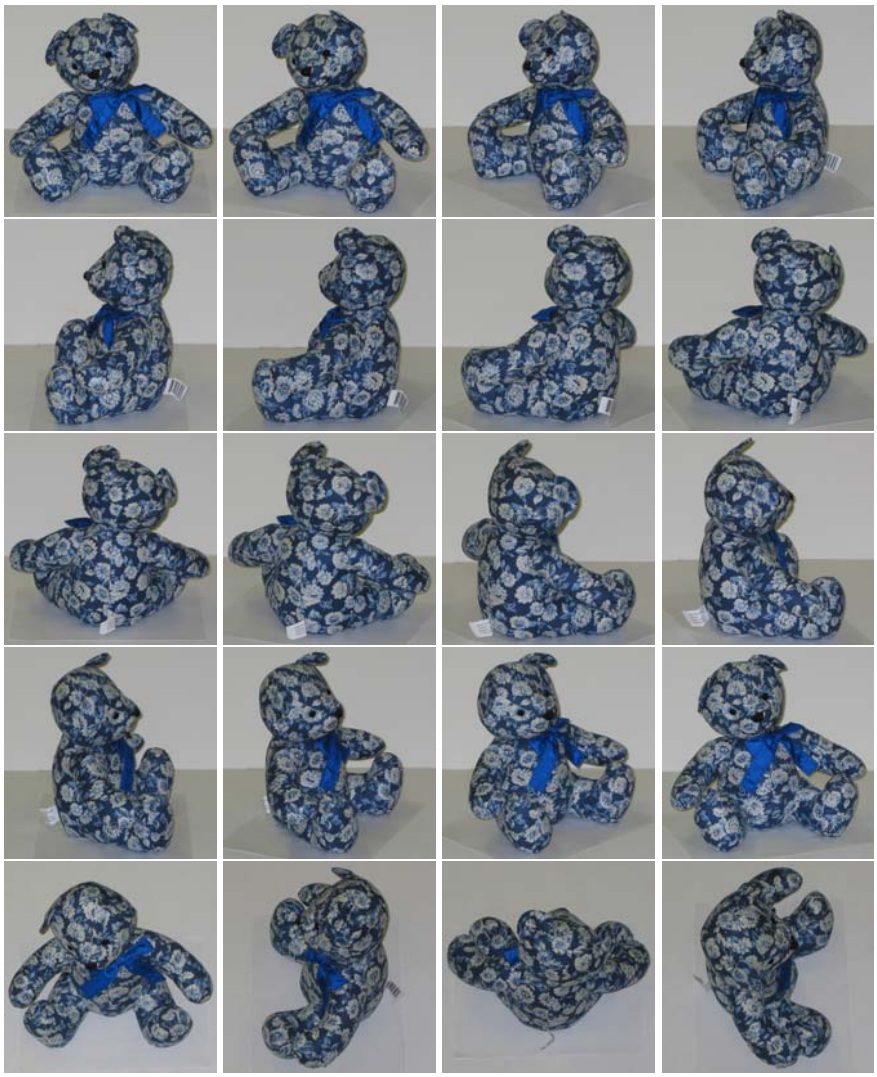


Figure 3.2: The 20 images used to construct the teddy bear model. There are 16 images roughly located in an equatorial ring, and 4 overhead images. This setup (with some variation in the number of input images) is typical of our modeling experiments.

### 3.2.1 Image Matching

As shown in Section 2.2, two images of two surface patches are sufficient to estimate the corresponding affine projection matrices and 3D patch configurations. Thus, all the power of the geometric constraints is available to guide image matching. Essentially, we combine wide-baseline stereo [7, 81, 84, 106, 113, 127, 135] with structure from motion [98, 129, 144].

While it is possible to select pairs of images to match from a set automatically [113], we have chosen to specify them manually using prior knowledge of the modeling setup: Typically, we acquire a number of views roughly located in an equatorial ring around the modeled object, as well as a couple of top and/or bottom views. Accordingly, we match pairs of successive equatorial images, plus some additional pairs where a top or bottom view has enough overlap with one of those from the ring.

After processing through point detectors and affine adaptation, an image can be viewed as simply a collection of affine regions. For each pair of images, we apply Algorithm 3 to match the two sets of regions. The remainder of this section gives implementation specifics for the algorithm in the context of image matching.

#### Appearance-Based Selection of Potential Matches

We do not use color information in modeling tasks, and rely exclusively on SIFT feature vectors to characterize local image appearance. A *match* is an ordered pair of patches, one from the model (i.e.: first) image and one from the test (i.e.: second) image. The initial list of potential matches is found by selecting for each patch in the model image the top $K$ patches in the test image as ranked by SIFT distance. In our experiments, $K$ is typically set to 5, which gives good results over all the objects. For objects with less distinctive texture (e.g.: the apple and the truck) it is useful to set $K$ to 10, which gives a richer set of matches. The cost of our (naive) implementation is $O(n^2 \log n)$, where $n$ is the number of affine regions found in the two images. Using efficient (and possibly approximate) algorithms for finding the $K$ nearest neighbors of a feature vector would obviously lower this cost, but this turns out to be negligible compared to the overall cost of Algorithm 3. Candidate matches whose SIFT feature vectors are separated by a Euclidean distance greater than $0.5$ are rejected.

For efficiency's sake, a simple neighborhood constraint is then used to further prune inconsistent matches: For a *primary* correspondence between image regions $R_m$ and $R_t$ to be retained, a sufficient fraction of the 10 nearest neighbors of $R_m$ should also match neighbors of $R_t$. Call the number of these *secondary* matches the *score* of the primary

correspondence they support. Since every affine region has roughly $K$ potential matches, the score is bounded by $10K$. We retain correspondences whose score is at least two standard deviations above average. In a typical case (matching the first two bear images), the mean score is 1.2, with a standard deviation of 3.1. The threshold for retaining matches is thus 7.4, and 1,150 of the initial 16,800 correspondences are retained in this case.

### Patch Refinement

The surviving matches go on to the robust estimation step (that is, Step 2 of Algorithm 3). This step and the subsequent expansion step both use matches to estimate the geometry of the scene. For that process to be reliable, matching rectified regions should line up as well as possible despite the unavoidable imperfections of affine adaptation in real images. It is therefore desirable to adjust the parameters of one of the rectified regions to maximize correlation with its match. Appendix B presents a simple non-linear least-squares solution to this problem (see [47, 120] for related approaches). Figure 3.3 shows an example. After refinement, only patch pairs whose normalized correlation is greater than $0.9$ are actually considered.

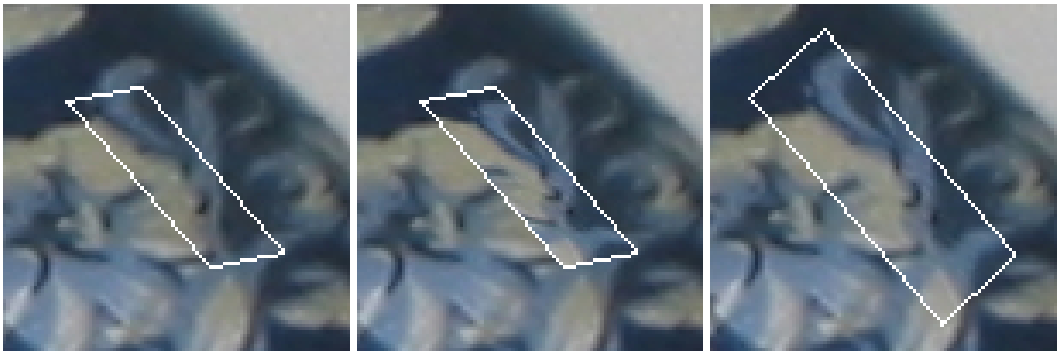


Figure 3.3: Adjusting the parameters of an affine region after matching. All three images are the same, except for the content and shape of the patch. Left: One of the affine regions in its original state. Middle: The texture inside the parallelogram is replaced by a matching region in a second image. Note that it does not register well with the surrounding texture. Right: Adjustment result. Note that the adjustment procedure is illustrated here in the original image domain, but the actual computations take place in the rectified domain.

### Robust Estimation

As discussed in Section 2.3, sampling and consensus are the key elements needed to implement RANSAC-like robust estimation. During sampling, factorization is used to solve

Eq. (2.1) for the two projection matrices and the two sample patches' configurations. During consensus, the projection matrices are held constant, and the configuration of every 3D patch is estimated from its matched pair of 2D patches using Eq. (2.1) via linear least squares. Those patches with low reprojection error are added to the consensus set.

Similar approaches have of course been used before in the context of wide-baseline stereo, although the geometric constraints exploited in that case are usually related to the distance between matching points and the corresponding epipolar lines [7, 81, 106, 113, 127, 135]. The reprojection error is a more natural metric in our context where two matching patches determine both the projection matrices and the 3D patch configurations, and it yields excellent results in practice.

In our experiments, we have used both plain RANSAC and a variant where the samples are chosen in a deterministic, greedy fashion. Concretely, the greedy variant uses each potential match as a seed for a group, iteratively adding the match minimizing the mean reprojection error until this error exceeds 0.1 pixels, or the group's size exceeds 20. In practice, both methods give almost identical results, RANSAC being slightly more efficient, and its greedy variant being slightly more reliable. The parameters used in our experiments are given in Figure 3.4, along with the computational costs for the two variants.

| Method | Cost | $K$ | $M$ | $N$ |
|---|---|---|---|---|
| RANSAC | $O(M\|P\|)$ | [5,10] | 1199 | 2 |
| Greedy | $O(N\|P\|^2)$ | [5,10] | $\|P\|$ | $\leq 20$ |

Figure 3.4: Parameters for the two robust estimation strategies used to match pairs of images in our experiments, along with their combinatorial cost. Here $|P|$ denotes the size of the set $P$ of match hypotheses, $K$ is the number of best matches kept per model patch, $M$ is the number of samples drawn, and $N$ is the size of one seed. The value of $M$ for RANSAC is based on an inlier rate of $w = 5\%$, $M$ being chosen in this case as $E(M) + 2S(M)$, where $E(M) = w^{-N}$ is the expected value of the number of draws required to get one good sample and $S(M) = \sqrt{1 - w^N}/w^N$ is its standard deviation. See [43, p. 347] for details.

We use a second neighborhood constraint to remove outliers at the end of this stage. It involves finding the five closest neighbors of a point in one image and the five closest neighbors of its putative match in the other image. If the match is consistent, the neighbors should also be matched with each other (barring occlusion). We test for this by comparing the barycentric coordinates of the centers of matched regions relative to all $\binom{5}{3} = 10$ triples of their neighbors (Figure 3.5). Barycentric coordinates are triples of numbers indicating the location of a point as a linear mixture of three reference points, and are invariant under affine transformations. The test is done symmetrically for the two images, and it examines 20 triples of neighbors. Two vectors of barycentric coordinates $\boldsymbol{x}$ and $\boldsymbol{y}$ are judged consis-

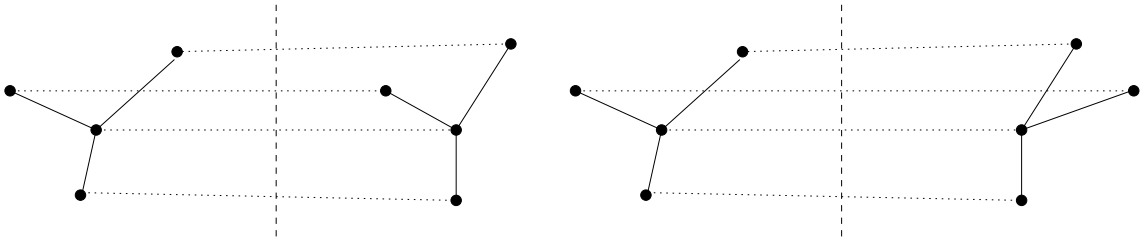


Figure 3.5: The barycentric neighborhood constraint. Left: Consistent matches. Right: Inconsistent ones.

tent if their relative distance $|\boldsymbol{x} - \boldsymbol{y}|/\max(|\boldsymbol{x}|, |\boldsymbol{y}|)$ is less than 0.5, and matches consistent with fewer than 8 of the 20 possible triples are rejected.

#### Geometry-Based Addition of Matches

The set of consistent matches found by the robust estimation stage typically provide a good estimate of the epipolar geometry of the image pair. Regardless of whether we are using the affine or the locally-affine (globally perspective) construction, we always estimate a projective fundamental matrix. For each patch in the model image, we search for all patches in the test image whose "epipolar distance" is less than 2.5 pixels. Specifically, we define the epipolar distance as $d(\boldsymbol{c}_m, \mathcal{F}\boldsymbol{c}_t) + d(\boldsymbol{c}_t, \mathcal{F}^T\boldsymbol{c}_m)$, where $d(\boldsymbol{p}, \boldsymbol{l})$ gives the perpendicular distance between a point $\boldsymbol{p}$ and a line $\boldsymbol{l}$ in pixels, $\boldsymbol{c}_m$ and $\boldsymbol{c}_t$ are the patch centers in the two images, and $\mathcal{F}$ is the fundamental matrix. We only add the nearest $K$ matches associated with a model patch in any given iteration of the expansion step of Algorithm 3.

### 3.2.2 Constructing an Integrated Model

The result of the image matching process is a collection of matches between neighboring training images (Figure 3.6). There are several combinatorial and geometric problems to solve in order to convert this information into a 3D model. The overall process is divided into four steps: (1) *chaining*: link matches across multiple images; (2) *stitching*: solve for the affine structure and motion while coping with missing data; (3) *bundle adjustment*: refine the model using non-linear least squares; and (4) *Euclidean upgrade*: use constraints associated with (partially) known intrinsic parameters of the camera to turn the affine reconstruction into a Euclidean one. The following sections describe each of these steps in detail.

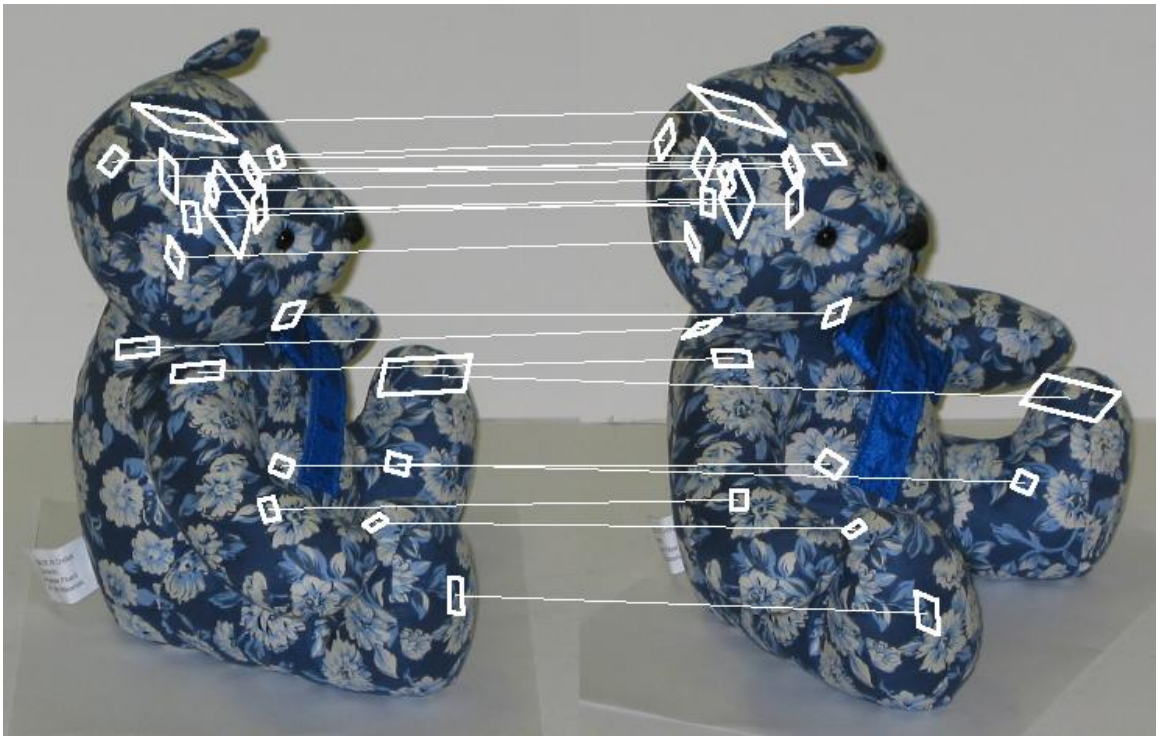


Figure 3.6: Matches between two images of the bear. For clarity, only 20 are shown.

**Chaining**

The matching process described in the previous section outputs pairs of affine regions matched across pairs of views. These pairs can be represented in a single *match graph* structure, where each vertex corresponds to an affine region, labeled by the image where it was found, and arcs link matched pairs of regions. Intuitively, the set of views of the same surface patch forms a connected component of the match graph, which can in turn be used to form a sparse *patch-view* matrix whose columns represent surface patches, and rows represent the images in which they appear (Figure 3.7).

Figure 3.7: A (subsampled) patch-view matrix for the teddy bear. The full patch-view matrix has 4,212 columns. Each black square indicates the presence of a given patch in a given image.

The measurements for a patch in all images where it appears must be self-consistent,

in the sense that the image measurements describe projections of exactly the same patch in space. It is not possible to directly enforce this because all we can measure is the projected texture of the patch. Instead we enforce the weaker condition of appearance consistency, in a similar manner to patch refinement when matching two images (Section 3.2.1). We do this in two steps. First we collate the results of pairwise refinement into an estimate of all the $\mathcal{S}_{ij}$ for a given surface patch $j$. Then we refine these estimates with respect to one reference patch.

The result of refinement between two patches is a pair of image measurement matrices $\mathcal{S}_f$ and $\mathcal{S}_v$, where $\mathcal{S}_f$ was kept fixed and $\mathcal{S}_v$ was modified by Levenberg-Marquardt (LM). Since matches are only refined on a pair-wise basis, it is possible for them to disagree on the value of a particular $\mathcal{S}_{ij}$. Therefore, we associate the affine transformation $\mathcal{H} = \mathcal{S}_f \mathcal{S}_v^{-1}$ (or equivalently $\mathcal{H} = \mathcal{S}_f \mathcal{R}_v$) with the edge of the match graph going from patch $v$ to patch $f$. The graph is undirected, so we must also associate $\mathcal{H}^{-1}$ with the edge from patch $f$ to patch $v$. Suppose that we know the matrix $\mathcal{S}$ for some node in the graph. We can estimate a consistent value for an adjacent node by finding the product $\mathcal{H}\mathcal{S}$ or $\mathcal{H}^{-1}\mathcal{S}$, depending on the direction along the edge.

For each connected component in the match graph, we select the patch with the largest scale as the reference. We think of this reference patch as the "root" node of its connected component. We then propagate the image measurements from it to all other connected patches. After each patch receives the propagated information, we again use LM to refine its estimated parameters with respect to the root patch.

In practice, the construction of the patch-view matrix is complicated by the fact that different paths may link a vertex of the match graph to more than one vertex associated with a single view. We have chosen a simple heuristic to solve this problem: after refining the parameters among all the patches in a connected component, we enumerate all the vertices associated with each image in the dataset, retain the representative vertex closest in feature space to the root vertex, and discard all others. This ensures that every image is represented by at most one vertex in each connected component.

### Stitching

The patch-view matrix is comparable to the data matrix used in factorization approaches to affine structure from motion [129]. If all patches appeared in all views, we could indeed factorize the matrix directly to recover the patches' 3D configurations as well as the camera positions. In general, however, the matrix is sparse, and we must find dense blocks (submatrices) to factorize and stitch. The problem of finding maximal dense blocks of views and

Figure 3.8: Refining patch parameters across multiple views: Rectified patches associated with a match in four views before (top) and after (bottom) applying the refinement process. The patch in the rightmost column is the "root", and is used as a reference for the other three patches. The errors shown in the top row are exaggerated for the sake of illustration: The regions shown there are the unprocessed output of the affine region detector. In actual experiments, the refined parameters found during image matching are propagated along the edges of the match graph to provide better initial conditions.

patches within the matrix reduces to the NP-complete problem of finding maximal cliques in a graph. Instead of solving this problem, we use the simple heuristic strategy given by Algorithm 4. It is not guaranteed to be optimal or complete, but generally produces an adequate solution. Briefly, we find a dense block for each patch—that is, for each column in the patch-view matrix—by searching for all other patches that are visible in at least the same views. In practice, this strategy provides both a good coverage of the data by dense blocks, and an adequate overlap between blocks. Typically, patches appear in at least three or four views, depending on the separation between successive views in the sequence, and there are in general two orders of magnitude more patches than views.

**Input:** For each patch $i$, a set $V_i$ of all views it appears in.
**Output:** A set of dense blocks of views $\times$ patches.

```
for all patches i do
  if no block has yet been seeded with a set of views equal to V_i then
    Seed a new block with views V_i.
    for all patches j do
      If V_i ⊆ V_j, then add patch j to the block.
    end for
  end if
end for
```

**Algorithm 4:** Find dense blocks.

The factorization technique described in Section 2.2 can of course be applied to each dense block to estimate the corresponding projection matrices and patch configurations in some local affine coordinate system (Figure 3.9). The next step is to combine the individual reconstructions into a coherent global model, or equivalently register them in a single coordinate system. With a proper set of constraints on the affine registration parameters, this can easily be expressed as an eigenvalue problem. In our experiments, however, we have found this linear approach to be numerically ill behaved (this is related to the inherent affine *gauge ambiguity* of our problem, see [132] for a discussion of this issue). Thus, in practice, we pick an arbitrary block as *root*, and iteratively register all others with this one using linear least squares, before using a non-linear method to refine the global registration parameters.

We use the *stitch graph* to assist in this process. Its vertices are the blocks, and an edge between two vertices indicates that the corresponding blocks overlap. We choose the largest block as root node and use its coordinate system as the global frame. We then find the best path from the root to every other node using a measure that maximizes the number of points shared by adjacent blocks, the rationale being that large overlaps will give reliable estimates of the corresponding (local) registration parameters. Specifically, we assign to each edge a *capacity* (the number of points common to the blocks associated with the incident vertices), and use a form of Dijkstra's algorithm to find for each vertex the path maximizing the capacity reaching the root.

The local registration parameters are concatenated along these paths, and they provide an estimate of the root-to-target affine transformation. Non-linear least-squares are finally used to minimize the mean-squared Euclidean distance between the centers of every pair of overlapping patches. After registering the blocks as described above, we combine all the camera and patch matrices into a single model. Since several blocks may provide a value for a given camera or patch, we give preference to those closer to the root.

Given that the cost of non-linear registration grows as $O(n^3)$ in the the number of edges in the stitch graph, it is useful to remove ("cull") some of the edges from large graphs. Algorithm 5 gives a procedure for doing this. It assumes that each node in the stitch graph has a pointer to its parent in the single best path back to the root. The idea behind the algorithm is to retain the best dense blocks (vertices) and enough of their overlaps (edges) to register them well. Each 3D patch is covered by one or more blocks, and we would like to retain the largest one. Since many 3D patches may share the same blocks, in general there will be fewer blocks than patches. Each block needs to overlap some other block that is registered with the root, so every edge on a path from some vertex back to the root is retained. Finally, some amount of redundancy in the paths back to the root improves

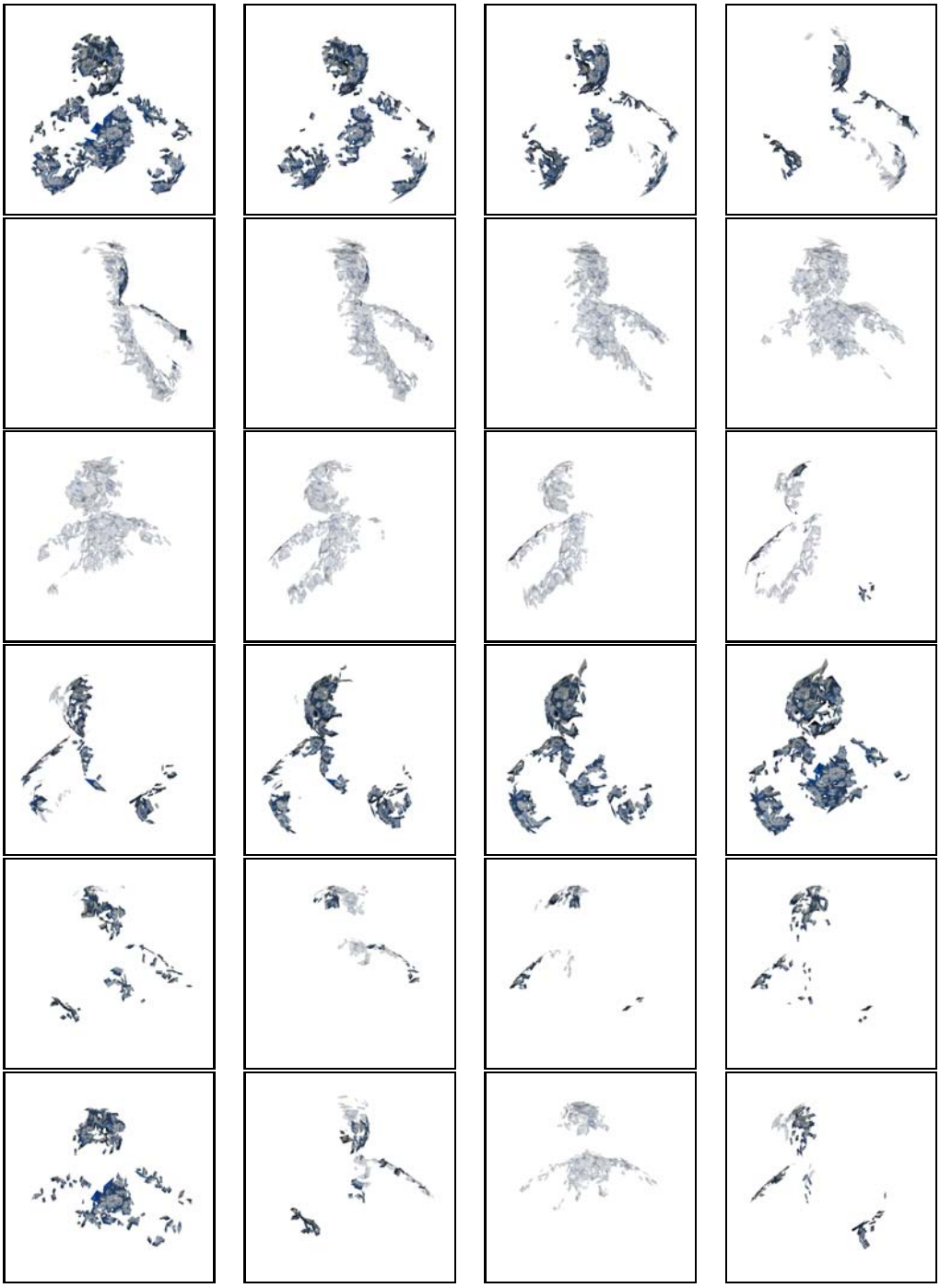


Figure 3.9: Sample partial models of the bear estimated from dense blocks. The blocks in this illustration were found by taking adjacent modeling views and selecting all patches they have in common. The partial models are all presented in a common coordinate frame, rather than in their local frames determined by factorization.

the quality of registration, so the algorithm retains a limited number of additional edges associated with each vertex.

**Input:** The stitch graph $G(V, E)$, along with parent information for each vertex so that it knows the one best path back to the root.
A minimum number $N$ of edges to keep for each retained vertex.
**Output:** The stitch graph $G(V, E)$ with some vertices and edges removed.

*Each vertex or edge may be either "marked" or "unmarked".*

- Set all vertices and edges to the unmarked state.
- Mark the largest block/vertex associated with each patch. The root vertex is also marked.
- For each marked vertex other than root, mark the $N$ edges with highest capacity.
- For each marked edge, ensure that both vertices are marked.
- For each marked vertex, follow the path back to the root, marking every edge and vertex along the way.
- Remove all unmarked vertices and edges.

**Algorithm 5:** Cull the Stitch Graph.

### Bundle Adjustment

Once all blocks are registered, the initial estimates of the variables $\mathcal{M}_i$ and $\mathcal{N}_j$ are refined by minimizing

$$E = \sum_{j=1}^{n} \sum_{i \in I_j} |\mathcal{S}_{ij} - \mathcal{M}_i \mathcal{N}_j|^2, \tag{3.1}$$

where $I_j$ denotes the set of images where patch number $j$ is visible. Given the reasonable guesses available from the initial registration, this non-linear least-squares process only takes (in general) a few iterations to converge.

We have implemented two non-linear methods for minimizing the error $E$ in Eq. (3.1). One is a sparse version of the Levenberg-Marquardt (LM) algorithm [87, 132]. The other uses the bilinear alternation strategy given by Algorithm 2, with appropriate equations. Note that the alternation strategy has first-order convergence properties. Since LM has second-order convergence [132], it will (in general) require fewer iterations. However, the cost for one LM iteration is much higher than one bilinear iteration. In practice we prefer the bilinear method, as for most problems it tends to finish much sooner and produces essentially the same results as sparse LM.

The completed 3D model (Figure 3.10) consists of the matrices $\mathcal{M}_i$ and a description of each 3D surface patch $j$: the matrix $\mathcal{N}_j$ and the corresponding rectified texture patch.

This patch can be constructed in a number of ways:

- Combine the texture information from each measured image patch into a single high-quality copy using super-resolution techniques [6, 16, 22], provided the patches satisfy our assumption of planarity and that they are well registered.
- Use PCA to determine an average patch and a small set of basis patches to represent the variability in the appearance [91, 133].
- Store multiple exemplar patches to cover the range of appearance variation.

The amount of variation in patch appearance is limited by the correlation thresholds used during modeling to link various views of the patch together. Currently, we simply choose the image patch with the largest characteristic scale and copy its texture into the model. This is sufficient for the purpose of matching the model to novel images, as the correlation threshold used there is never tighter than the threshold used for modeling.

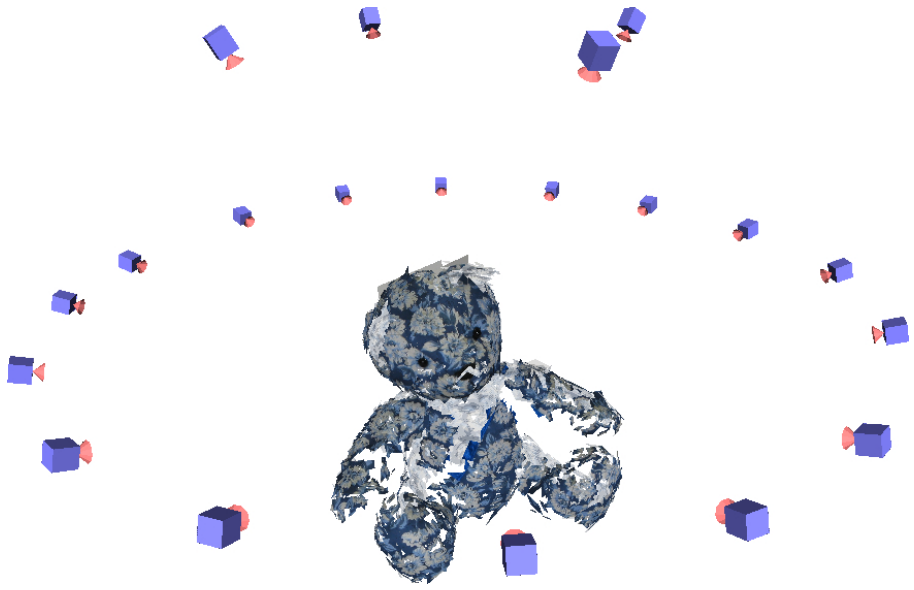


Figure 3.10: The bear model, along with the recovered affine camera configurations.

**Euclidean Upgrade**

It is not possible to go from affine to Euclidean structure and motion from two views only. When three or more views are available, on the other hand, it is a simple matter to compute the corresponding Euclidean weak-perspective projection matrices (assuming zero skew and known aspect-ratios) and recover the Euclidean structure [103, 129]: Briefly, we find

the $3 \times 3$ matrix $\mathcal{Q}$ such that $\mathcal{A}_i \mathcal{Q}$ is part of a (scaled) rotation matrix for $i = 1, \ldots, m$. This provides linear constraints on $\mathcal{Q}\mathcal{Q}^T$, and allows the estimation of this symmetric matrix via linear least-squares. The matrix $\mathcal{Q}$ can then be computed via Cholesky decomposition for example [98, 144].

### 3.2.3 Experimental Results

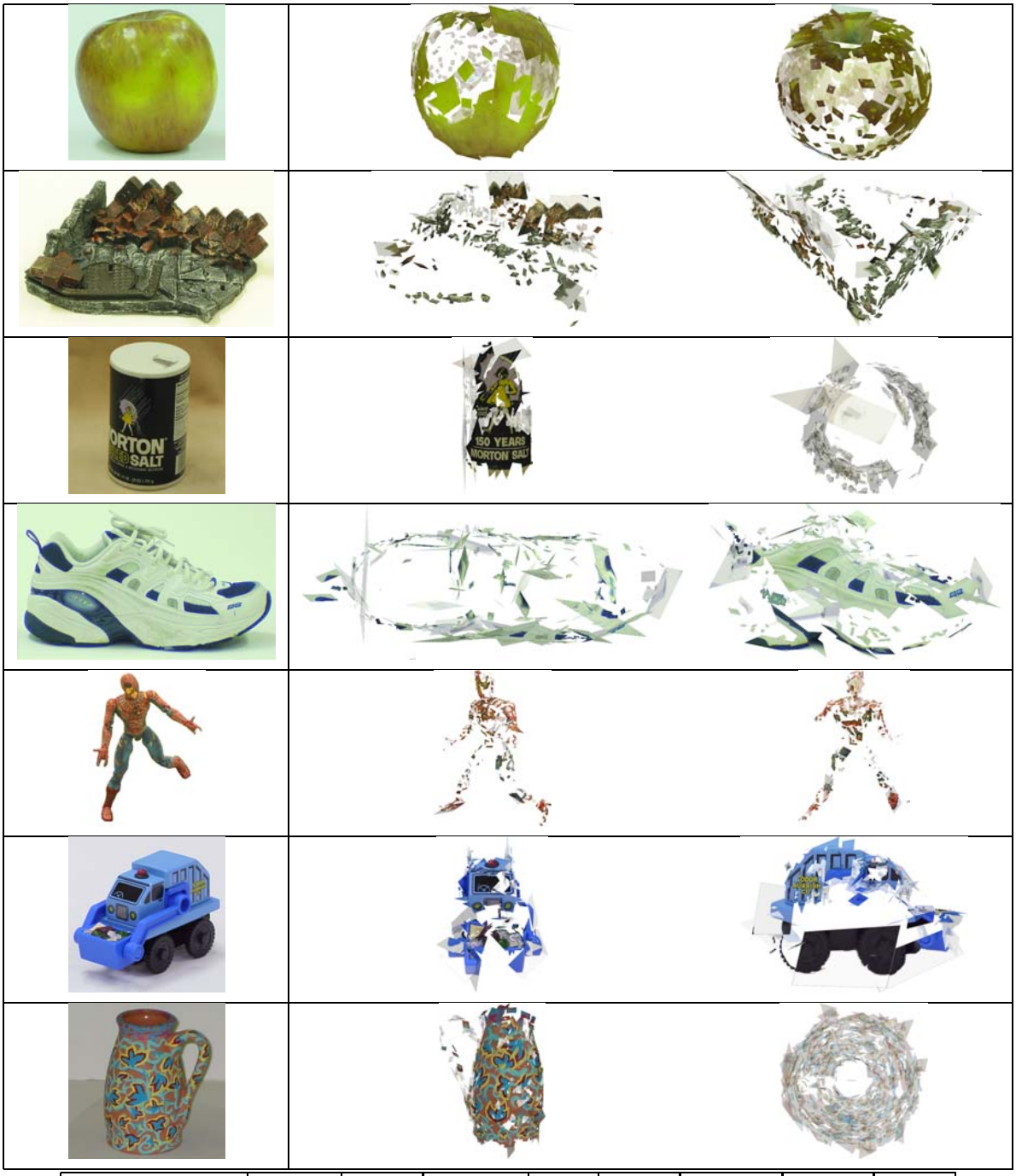
The current implementation of our modeling approach is quite reliable, but rather slow: The teddy bear shown in Figure 3.10 is our largest model, with 4014 model patches computed from 20 images (24 image pairs). Image matching takes about 75 minutes per pair using pure RANSAC, for a total of 29.9 hours.[1] Image matching using the greedy algorithm takes 88 minutes per pair for a total of 35.2 hours. The final model is assembled from the partial ones in 1.5 hours. The greatest single expense in our modeling procedure is patch refinement. By selecting less stringent convergence criteria for this process and using a fixed $16 \times 16$ resolution for the image regions used to drive the LM procedure, it is possible to reduce the matching time to 6.6 minutes per image pair and assemble the model in 42 minutes, at the cost of getting 4% fewer 3D patches. Since modeling speed is not a priority in the context of this presentation, we have used the original refinement parameters in the rest of our experiments.

We have applied the modeling approach presented in this section to seven other objects, namely: an apple, the rubble-covered stand for a Spiderman action figure (called simply "rubble" from now on), a salt can, a shoe, Spidey himself, a toy truck, and a vase (Figure 3.11). For each object, the figure shows one sample from the set of input pictures. Each object model has been constructed using 16 to 20 input images, except for the apple which is modeled from 29 images to attain complete surface coverage. Beside each sample input image, the figure shows two renderings of the recovered Euclidean model. The models are rather sparse, but one should keep in mind that they are intended for object recognition, not for image-based rendering applications.

## 3.3 Recognition

We now assume that the modeling approach presented in Section 3.2 has been used to create a library of 3D object models, and address the problem of identifying instances of these models in a test image. In many respects, this process is analogous to the method

[1] All computing times in this presentation are given for C++ programs executed on a 3Ghz Pentium 4 running Linux.




| | Apple | Bear | Rubble | Salt | Shoe | Spidey | Truck | Vase |
|---|---|---|---|---|---|---|---|---|
| Input images | 29 | 20 | 16 | 16 | 16 | 16 | 16 | 20 |
| Model patches | 759 | 4014 | 737 | 866 | 488 | 526 | 518 | 1085 |

Figure 3.11: Object gallery. Left column: One of several input pictures for each object. Right column: Renderings of each model, not necessarily in same pose as input picture. Top to bottom: An apple, rubble (Spiderman base), a salt can, a shoe, Spidey, a toy truck, and a vase.

described in Section 3.2.1 for pairwise image matching. As before, Algorithm 3 outlines the overall process. Further details are given in the rest of this section.

### 3.3.1 Appearance-Based Selection of Potential Matches

Since matching is much more challenging in the recognition context where images may be heavily cluttered than in modeling tasks where there is essentially no clutter, we exploit both the SIFT descriptors and color histograms to select initial matches. More specifically, we use (1) a measure of the contrast (average squared gradient norm) in the patch, (2) a $10 \times 10$ color histogram drawn from the UV portion of YUV space, and (3) SIFT. To match feature vectors, we rely on color to filter out unpromising matches before comparing the remaining ones with SIFT. The level of contrast determines whether to use a tight or relaxed threshold on color.

We compare color histograms with the $\chi^2$ metric, defined as

$$\sum_i \frac{(a_i - b_i)^2}{a_i + b_i},$$

where $a_i$ and $b_i$ are corresponding bins of the two histograms. The resulting value is in $[0, 2]$, with 0 being perfect match and 2 being complete mismatch.

Figure 3.12 illustrates the usefulness of multiple local image descriptors in matching tasks, particularly when the patches have low contrast. This example is taken from a test image for the apple. The model patch is in the center, the correct match is on the left, and an incorrect match is on the right. By human perception, all three patches appear almost identical, except that the incorrect patch has a different color. By SIFT distance, the incorrect match is actually closer than the correct one. The use of a color descriptor enables us to select the correct match.

We use as before non-linear least squares to refine the parameters of the matched image regions to maximize their correlation with the corresponding model patches. Since this process is computationally expensive, we first apply a neighborhood constraint similar to that used in image matching to discard obviously inconsistent matches, as described next.

#### Euclidean Neighborhood Constraints

We saw earlier that affine models constructed from multiple views can be upgraded into Euclidean ones. In turn, a Euclidean model can be used to impose neighborhood constraints on individual matches: It is well known that three point matches—or in our case,

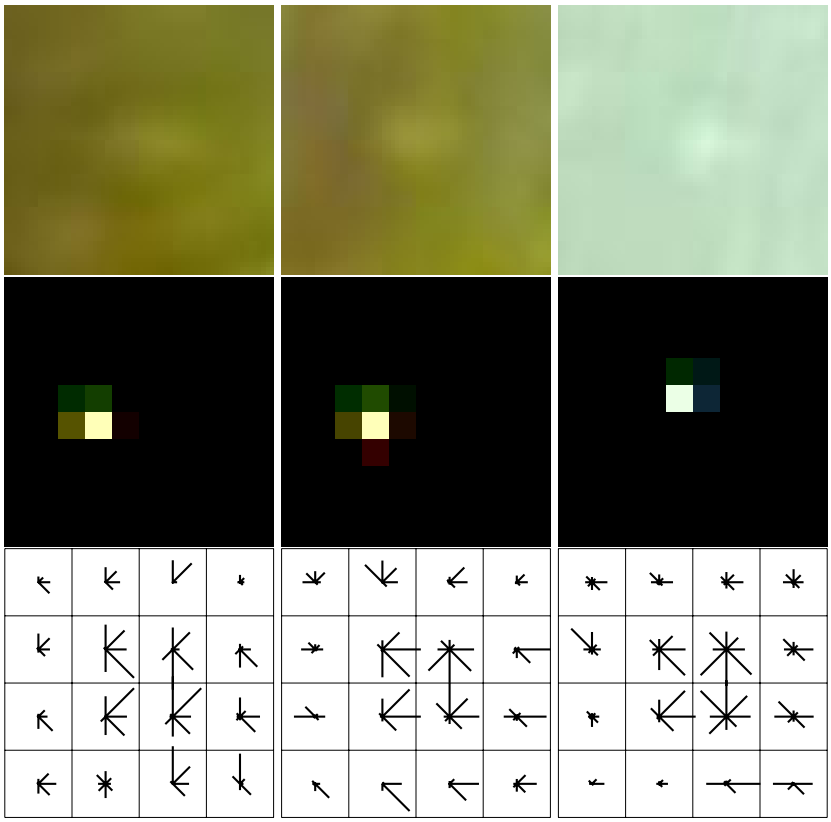


Figure 3.12: Comparing SIFT and color descriptors on low-contrast patches. The center column is the model patch. The left column is the correct match in the image. The right column is the match in the image ranked first by SIFT (but that is in fact an incorrect match). The top row shows the patch and the bottom row shows the SIFT descriptor. The middle row shows the color histogram in the form of a grid of colored blocks, where the brightness of the block indicates the weight on that color. The incorrect match has a Euclidean distance of 0.52 between SIFT descriptors and a $\chi^2$ distance of 1.99 between the corresponding color histograms; and the correct match has a SIFT distance of 0.67 and a color distance of 0.03. The two patches on the left are red-green colored, while the patch on the right is aqua.

a single match between the corners and center of a model patch and those of an affine image region—are sufficient to determine the pose of a 3D object for calibrated cameras [56]. Thus, we recover the object pose associated with each potential match, and use it to reproject all other model patches into the image. Any patch whose reprojection falls close enough to a compatible affine region casts a vote for the match. Match candidates with above-average support are retained, and passed on to the refinement step.

In our implementation, the weight $w$ of each vote depends on three factors, namely the characteristic scale $\sigma_0$ of the *primary* image region associated with the match candidate, the distance $d$ between the projection of the voting patch and the corresponding *secondary* image region, and the distance $d_0$ between the primary and secondary regions. In practice, we set $w = G_\sigma(d)$, where $G_\sigma$ is a Gaussian distribution with standard deviation $\sigma = 10 + d_0/4\sigma_0$ (Figure 3.13). With this choice, small values of $d$ correspond to large votes, and the contribution of each secondary patch is modulated so the Gaussian sharply peaks for large primary regions likely to yield accurate pose estimates and for secondary regions close to the primary ones that are more likely to be accurately localized.

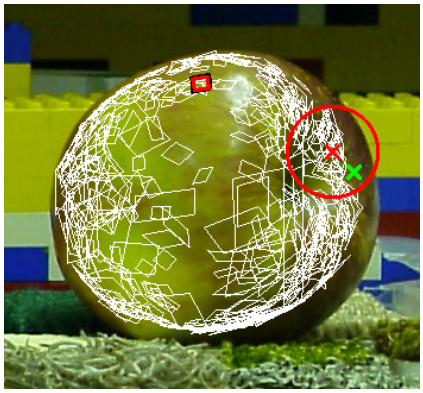


Figure 3.13: An illustration of the proposed voting scheme: The primary match that determines the pose appears as a heavy parallelogram, and all the forward facing patches projected from the model appear as light parallelograms. The projected center of the supporting match appears as an "$\times$" surrounded by a circle. The actual image position of the supporting match appears as another "$\times$". The radius of the circle is equal to the standard deviation of the Gaussian distribution deciding the weight of the corresponding vote.

### 3.3.2 Estimating Geometry

As noted in Chapter 2, various methods for finding matching features consistent with a given set of geometric constraints have been proposed in the past, including interpretation tree (or alignment) techniques [5, 33, 49, 56, 69], geometric hashing [61, 62], and robust statistical methods such as RANSAC [36] and its variants [130]. Both alignment and

RANSAC can easily be implemented in the context of Algorithm 3. We have experimented with several alternatives: The first one is a recursive implementation of alignment where an interpretation tree is visited in a depth-first manner (*null* matches between model patches and "empty" image regions being used to handle occlusion and faulty detection) until a maximum depth $N$ is reached ($N = 20$ in our experiments), or the mean reprojection error exceeds 1 pixel in all branches up to that depth (see Ayache and Faugeras [5], Faugeras and Hebert [33], and Grimson and Lozano-Pérez [49] for more details on this approach).

We have also implemented plain RANSAC; a "greedy" version where, as described in Section 3.2.1, $M$ groups of matches of size lesser than or equal to $N$ are chosen in a deterministic, greedy manner to minimize the mean projection error, and used instead of random samples; and an "exhaustive" version where all pairs of candidate matches are examined. The computational costs of the RANSAC variants are easy to estimate, and they are given in Figure 3.14. The cost of alignment is more difficult to assess, but can be shown to be a low-order polynomial in the size $n$ of the model when there is little or no clutter, and exponential in $n$ in the presence of clutter when no limit on the depth of the tree search is imposed [48]. The worst-case computational complexity of our bounded tree search is clearly polynomial, and bounded above by $O(n^N)$, but determining its *expected* cost is beyond the scope of this thesis. As will be shown in Section 3.3.5, the "greedy" version of RANSAC has performed best in our experiments.

| Method | Cost | K | M | N |
|---|---|---|---|---|
| RANSAC | $O(M\|P\|)$ | $L/n$ | $[1998, 12498]$ | 2 |
| Alignment | see Sec. 3.3.2 | $L/n$ | $n$ | 20 |
| Exhaustive | $O(\|P\|^3)$ | $L/n$ | $\|P\|^2$ | 2 |
| Greedy | $O(N\|P\|^2)$ | $L/n$ | $\|P\|$ | 20 |

Figure 3.14: Parameters for the different geometric estimation methods for Algorithm 3 used in our recognition experiments, along with their combinatorial cost. Here, $L$ denotes a preset number of potential matches to be examined ($L = 12,000$ in our experiments), and $n$ is the number of patches per object model.

### 3.3.3 Geometry-Based Addition of Matches

The matches found by the geometric estimation stage provide a projection matrix that places the model into the image. All forward facing patches in the model could potentially be present in the image. Therefore, we project each such patch into the image and select the $K$ closest image patches as new match hypotheses.

### 3.3.4 Object Detection

Once an object model has been matched to an image, some criterion is needed to decide whether it is present or not. After experimenting with a few reasonable choices, we have settled on the following criterion:

$$(\text{number of matches} \geq m \text{ OR matched area/total area} \geq a) \text{ AND distortion} \leq d,$$

where nominal values for the parameters are $m = 10$, $a = 0.1$, and $d = 0.15$. Here, the measure of distortion is

$$\frac{\boldsymbol{a}_1^T \boldsymbol{a}_2}{|\boldsymbol{a}_1||\boldsymbol{a}_2|} + \left(1 - \frac{\min(|\boldsymbol{a}_1|, |\boldsymbol{a}_2|)}{\max(|\boldsymbol{a}_1|, |\boldsymbol{a}_2|)}\right),$$

where $\boldsymbol{a}_i^T$ is the $i$th row of the leftmost $2 \times 3$ portion $\mathcal{A}$ of the projection matrix, and it reflects how close to the top part of a scaled rotation this matrix is. The matched surface area of the model is measured in terms of the patches whose normalized correlation is above the usual thresholds, and it is compared to the total surface area actually visible from the predicted viewpoint. The influence of the three parameters on recognition performance is studied in the next section.

### 3.3.5 Experimental Results

Our recognition experiments match all eight object models against a set of 51 images (the photograph from Figure 3.1 and the 50 pictures shown in Figure 3.15). Each image contains instances of up to five object models, even though most of them only contain one or two. Figure 3.16 gives quantitative recognition results for the different monochrome variants of our algorithm, where color information is not used. The parameters for these tests are fixed to their nominal values of $m = 10$, $a = 0.1$, and $d = 0.15$. With these settings, none of the methods tested gives false positives, and the "greedy" version of RANSAC with $N = 20$ gives the best performance, with a recognition rate (averaged over the eight object models) of 88%. The time costs given in the table are per image-object combination, in minutes.

Since it has consistently performed best in our experiments, we will from now on focus on the greedy variant of RANSAC with $N = 20$. It is interesting to compare different image descriptors and to test whether the use of color information may boost recognition performance. Figure 3.17 shows the results of a quantitative experiment: It can be seen that the combination of color and SIFT gives the best performance, with a mean recognition rate

Figure 3.15: The dataset (51 images) used in our recognition experiments: 50 of the images are shown here. The last one is shown in Figure 3.1.

| Method | Apple 11 | Bear 11 | Rubble 9 | Salt 10 | Shoe 9 | Spidey 4 | Truck 12 | Vase 12 | Mean | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| RANSAC | 3 | 11 | 8 | 9 | 2 | 3 | 9 | 11 | 71% | 4.3 |
| Alignment | 5 | 10 | 9 | 10 | 4 | 4 | 12 | 12 | 85% | 7.5 |
| Exhaustive | 5 | 11 | 9 | 10 | 4 | 4 | 12 | 12 | 86% | 7.7 |
| Greedy ($N = 2$) | 6 | 11 | 9 | 10 | 3 | 4 | 12 | 12 | 86% | 5.9 |
| Greedy ($N = 20$) | 5 | 11 | 9 | 10 | 5 | 4 | 12 | 12 | 88% | 6.7 |

Figure 3.16: Comparison of recognition rates for different monochrome variants of our method. See text for details. The row of numbers immediately under the object names gives the true number of instances present in the test images.

of 94%, and only four out of 51 images where recognition fails. Using color together with plain patch correlation results in performance similar to that of SIFT descriptors without color information.

| Method | Apple 11 | Bear 11 | Rubble 9 | Salt 10 | Shoe 9 | Spidey 4 | Truck 12 | Vase 12 | Mean | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Correlation | 6 | 11 | 8 | 10 | 4 | 4 | 10 | 8 | 80% | 5.6 |
| SIFT | 5 | 11 | 9 | 10 | 5 | 4 | 12 | 12 | 88% | 6.7 |
| Correlation + Color | 8 | 11 | 9 | 10 | 6 | 4 | 10 | 11 | 89% | 3.9 |
| SIFT + Color | 8 | 11 | 9 | 10 | 7 | 4 | 12 | 12 | 94% | 3.7 |

Figure 3.17: Comparison of recognition rates for different descriptors using the greedy RANSAC variant with $N = 20$.

As is always the case in object recognition, many implementation parameters can be varied in our program: For example, Figure 3.18 shows the trade-off between computing cost and recognition accuracy that can be achieved by changing the patch size used to refine the alignment between matched affine regions. As shown by this figure, selecting a fixed $16 \times 16$ resolution instead of the original resolution of the test patch used in the previous experiments halves the computing time with essentially no effect on recognition accuracy. Lowering the resolution too much, on the other hand, clearly affects recognition performance.

The recognition rates reported so far are for fixed, nominal values of the detection parameters $m$, $a$, and $d$. A better understanding of our algorithm's performance can be

| Method | Apple 11 | Bear 11 | Rubble 9 | Salt 10 | Shoe 9 | Spidey 4 | Truck 12 | Vase 12 | Mean | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Original resolution | 8 | 11 | 9 | 10 | 7 | 4 | 12 | 12 | 94% | 3.7 |
| $16 \times 16$ resolution | 8 | 11 | 9 | 10 | 7 | 4 | 12 | 12 | 94% | 1.9 |
| $8 \times 8$ resolution | 9 | 11 | 9 | 10 | 5 | 4 | 11 | 12 | 91% | 1.6 |

Figure 3.18: Effect of region sampling during patch refinement on computation cost and recognition accuracy.

gained by plotting the overall rates of true positives (instances where an object is correctly identified in an image) and true negatives (instances where an object is correctly determined to be absent) against a range of parameter values. Figure 3.19 shows the corresponding plots for the color version of our algorithm, where we vary one of the three parameters while holding the other two constant at their nominal values.

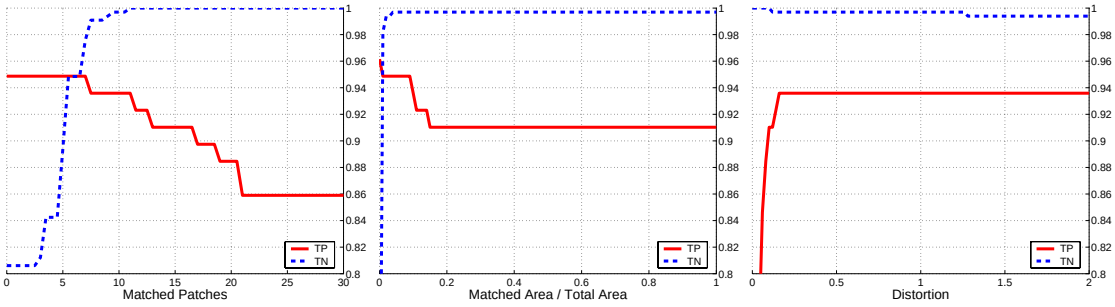



Figure 3.19: Dependency of the recognition rate on the detection parameters: The true positive (TP) and true negative (TN) rates are plotted by holding two of the detection parameters constant at their nominal values and varying, from left to right, the number of matched patches, the ratio of matched to visible area, and the distortion.

As shown by Figure 3.19, the recognition performance is quite stable over a reasonable range of detection parameters. The equal-error-rate parameter values correspond to the point (if any) where the true positive and true negative curves cross, with true positive rates in the 94-96% range in our experiments.

We believe it is important to evaluate recognition methods objectively and quantitatively. There is no common dataset for testing recognition performance, but an alternative is to evaluate each method on the datasets collected by other researchers. With that in mind, we asked several researchers to test their respective methods on our dataset, and likewise we tested our method on their datasets. Figure 3.20 shows the result of the comparison on our dataset (that is, the one presented in the recognition experiments above). The researchers listed in the legend of Figure 3.20 are: Ferrari et al. [35], Lowe [72], Mahamud & Hebert [76], and Moreels et al. [88]. Each of them performed the testing for their respective methods and then provided us with their results.

Lowe obtained essentially the same level of performance as our method. However, the exact set of objects on which his method fails is slightly different: three images of the apple and two images of the salt can. Ferrari's method also attains the same level of performance as ours, albeit with more false positives. It eventually recovers the correct pose for all the objects except one image of the salt can. These results suggest that recognition and pose recovery failures in the various recognition systems may be due as much to their

implementation details as to the characteristics of specific test shots. It is also interesting to note that the recognition systems tend to fall into two distinct curve patterns. Lowe's and ours are essentially "flat" at a high level across the whole ROC space, while the other methods tend to follow a curve that rises with false positives in a typical manner.

We also tested our system using only image matching rather than 3D models, with the results shown by the "wide-baseline matching" curve. Specifically, we matched each test image against all the training images using the method described in Section 3.2.1, and considered an object to be recognized if the number of matches between a test image and a training image exceeded a threshold. The tests demonstrate that 3D constraints improve recognition performance on our dataset.

On the other hand, our method did not succeed on any other dataset provided. They key issue is that our method requires multiple views of the same object instance to construct a model. Of the other datasets we tested, only the one provided by Vittorio Ferrari has a large number of views (up to 8) per object. However, they are so widely separated that it is difficult to find enough points in common across triples of views to assemble a global model. We are experimenting with modeling from stereo pairs without constructing a global model.

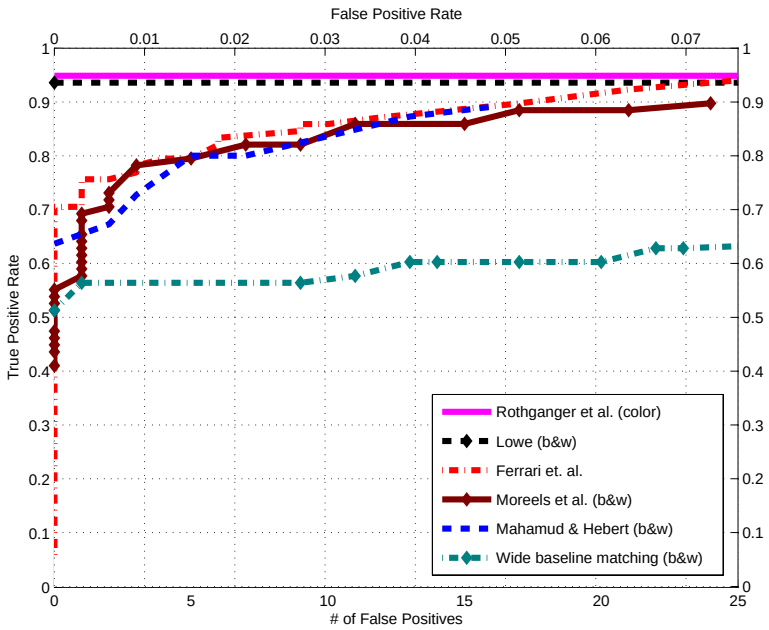



Figure 3.20: True positive rate plotted against number of false positives for several different recognition methods. For our curve, the three recognition parameters $m$, $a$, and $d$ assume their best values for each level of false positives.

Let us conclude with some qualitative experimental results, using as before the color

+ SIFT greedy variant of RANSAC with $N = 20$. Figure 3.21 shows sample results of some challenging yet successful recognition experiments, with a large degree of occlusion and clutter. Figure 3.22 shows closeups of the four cases where recognition fails. Very little of the apple is visible in the two images where our program fails to recognize it. Maybe more surprisingly, the shoe occupies a large portion of the two images where it escapes detection. The shoe images shown in Figure 3.22 are separated by about $60^\circ$ from the views used during modeling. This situation may be helped by having some overhead training images. Unfortunately, we only took equatorial training shots, and the object no longer exists in a usable form.

## 3.4 Discussion

*The main contribution of this chapter is an implemented algorithm for automatically acquiring 3D models of rigid objects from a sparse set of unregistered photographs and recognizing them in cluttered photographs taken from unconstrained viewpoints.* It uses invariants as a local object description by exploiting the fact that smooth surfaces can be approximated by sufficiently small planar patches. Combining this idea with the affine regions of Mikolajczyk and Schmid ([84]) has allowed us to construct a normalized representation of local surface appearance that can be used to select promising matches in 3D object modeling and recognition tasks. We have used multi-view geometric constraints to represent the larger 3D surface structure, retain groups of consistent matches, and reject incorrect ones. Our experiments demonstrate the promise of the proposed approach to 3D object recognition.

Admittedly, our current implementation is slow, especially compared to the systems proposed by Lowe [70], and Mahamud and Hebert [76], that achieve frame-rate object detection in cluttered scenes. Speed was never our priority (despite some efforts at optimizing our code), and we believe that our approach can (and should) be sped up by at least an order of magnitude using a more careful implementation. Two changes would greatly improve the performance: 1) Make the method more robust to the geometric error introduced by mis-estimated patch shape. This may allow the elimination of patch refinement in the recognition process. For modeling, it would still be important to refine the patches to get the most precise model possible. 2) Use a voting scheme to select the most promising models. The current implementation treats every model $\times$ image pair as a completely independent problem, rather than committing to a choice between available models in a single step.

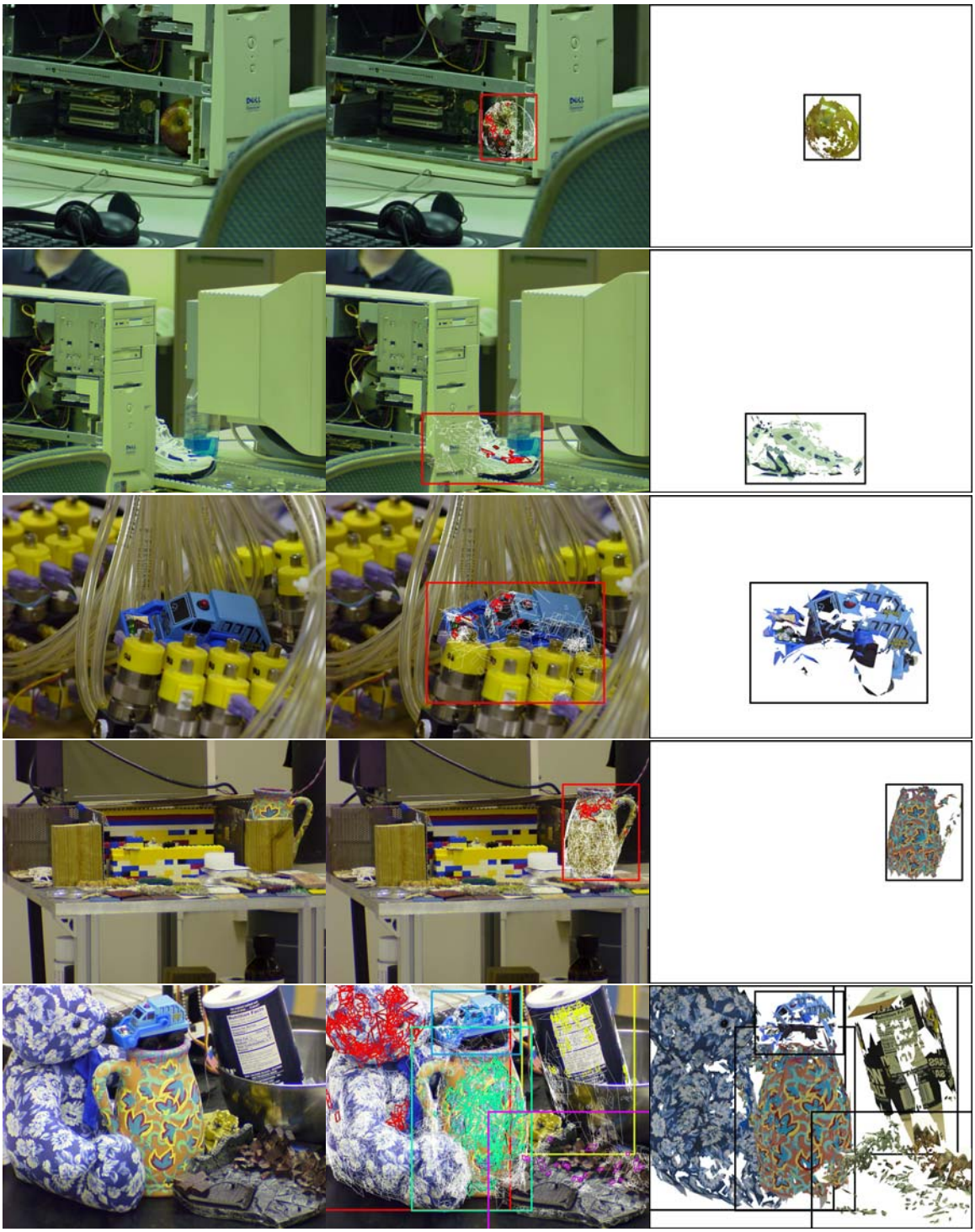


Figure 3.21: Some challenging, but successful recognition results. As in Figure 3.1, the recognized models are rendered in the poses estimated by our program, and bounding boxes for the reprojections are shown as rectangles.

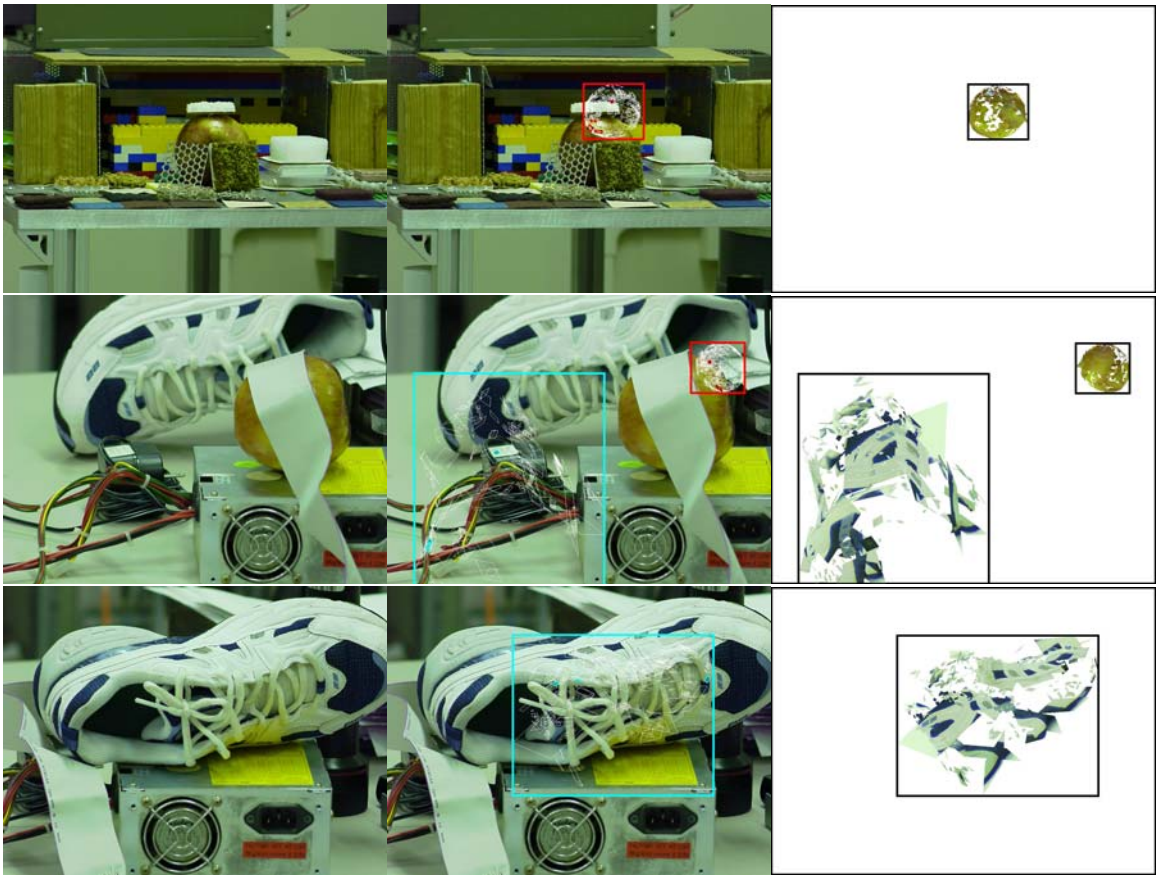


Figure 3.22: Closeups of the images where recognition fails.

# Chapter 4

# Image Sequences

This chapter addresses the problem of modeling and recognizing rigid components in image sequences that may contain multiple moving objects observed by a moving camera [13, 29]. The key insight is that the geometric consistency constraint derived in Section 2.2 also holds for the *rigid parts* of an articulated object. In Chapter 3, multi-view geometric constraints associated with affine-invariant patches guided the matching process and recovery the 3D object shape. Here, they serve as rigidity constraints, guiding motion segmentation and the recovery of scene structure.

We apply this approach to the problem of shot matching, demonstrating a simple system that matches 3D scene models constructed from different shots. Shot matching can be broadly defined as the video analog of image retrieval: given some model of a desired object or scene, return all shots that contain instances of that model. Video retrieval systems typically use appearance similarity measures to retrieve [20, 40, 46, 123] or to cluster shots [19, 113]. As in the case of wide-baseline matching, if they make use of geometric constraints, then those constraints tend to be 2D in nature. Many shots in typical movies tend to have nearly degenerate motion or dominant planes, making 3D information difficult to obtain [101]. However, when 3D structure is available, it provides inherently stronger matching constraints.

## 4.1 Background

The digital storage of video content became feasible during the 1990's thanks to the development of *discrete cosine transform* (DCT) based compression methods such as those used in JPEG and MPEG [50], making the enormous amount of data tractable. Simultaneously, the growth in computing power resulting from "Moore's Law" [86] enabled the processing of video in real time. This conjunction naturally encouraged the development of video

indexing systems (e.g., [20, 31, 63, 138]), which are analogous to image retrieval systems (e.g., [18, 40, 73, 83, 115]).

Automatic video analysis is a broad topic that includes indexing video as well as other activities such as surveillance and navigation. It can be roughly divided into three related areas:

1. Temporal segmentation – Dividing the video into shots and (sometimes) into larger units such as theatrical scenes and acts.

2. Object description – Characterizing the appearance and motion of components contained in the video.

3. Semantic description – High-level description of the behavior and affordances of objects.

We briefly discuss each of these areas of work below.

Temporal segmentation is generally a prerequisite for all other analysis, and the "shot" is accepted as the atomic unit of video for most purposes. A common definition is "one or more frames generated and recorded contiguously and representing a continuous action in time and space" [27]. Researchers have been working on shot segmentation for at least a decade, and while there remain some open problems, it is fairly mature [65, 95].

Object description addresses the contents of the video at a low level. For example, one may describe the appearance of regions of the video, or measure the optical flow field. Object description may follow the traditional path for still images [38, 40, 112]. However, video contains motion information that provides an opportunity for new types of descriptors. For example, the motion vector field in MPEG compressed video can itself be treated as a texture [74]. Some video retrieval systems track the motion of blobs and allow the user to search based on the approximate trajectories of these blobs [20, 149], or they may separate the background into a mosaic on which foreground objects move [31, 40].

Semantic descriptions such as "running water" or "person stealing stuff" are the level at which humans would prefer to query a video library. Despite the difficulty, work has progressed in this area [1, 21, 23, 46, 93], perhaps even more than in still images. This may be because semantic descriptions often have an action component, which is easier to infer from moving images.

To make use of the detected features, a video retrieval method requires a similarity measure. Because of its time component, video gives the opportunity for and sometimes demands more sophisticated measures than photographs. Many methods compare features without considering time (e.g., [40]). Some methods may compare features derived from

motion, but that is not necessarily the same as having a time component in the comparison method itself. Some comparison methods do exist, however, that incorporate a true temporal component. One approach is to to align keyframes in a pair of shots [126] or to align shots in the video as a whole [146]. Another method is to compare features in a space where time is either a (implicit or explicit) dimension of that space [20, 26] or of a subspace [148].

Our work falls mainly into the object description area, in that we extract the appearance and structure of rigid components in the video. We do not attempt to do any semantic analysis, and we assume that shot segmentation is given as part of the input.

## 4.2 Related Work

The approach presented in this chapter can be applied to indexing and retrieval of shots in video libraries. However, unlike most existing video retrieval methods, which only deal with 2D image motion, ours takes full advantage of strong 3D geometric constraints. In this section we briefly review related work in video analysis, then discuss relevant techniques for 3D object modeling from video sequences, and finish with a survey of motion segmentation methods.

### 4.2.1 Video Analysis and Shot Matching

Schaffalitzky and Zisserman [112] apply their method of photograph matching [113] to the problem of shot matching. They take as input a set of shots, with the keyframes within each shot identified. The goal is to cluster similar shots together. They measure the similarity between two shots (say, in a movie) by finding the number of point matches between every combination of a fixed number of keyframes in the two shots and then combine these into a single measure.

Sivic and Zisserman [123] recast this approach in a text retrieval context, which they dub "Video Google". The idea is to extract a vocabulary of SIFT descriptors by vector quantization and treat keyframes as documents containing instances of these "words". Each word is completely indexed, and retrieval is based on similarity of frequency vectors. They further rank retrieval results by applying a neighborhood constraint to the arrangement of patches between the query and test frames.

An alternative to addressing individual keyframes is to form a mosaic from the keyframes in the shot and match the mosaics [3, 31, 40]. For example, Aner and Kender [3] use "rubber sheet" matching between mosaics of shots to measure similarity. Rubber sheet match-

ing uses color histograms drawn from subregions of each mosaic to find what portions line up between the two. This is necessary since mosaics of the same scene but generated from different shots may have different width, alignment and scale.

### 4.2.2 Automated Acquisition of 3D Object Models from Image Sequences

The video indexing methods described above only recover the 2D motion of image regions, and do not take advantage of the strong geometric constraints that can be derived for rigid bodies observed by an affine or perspective camera. By contrast, rather than match 2D structures found in the frames of a shot, we propose to approach the problem of shot matching by forming 3D reconstructions of the contents of the shots and match these reconstructions. The traditional approach to 3D modeling from image sequences has been point-match based structure-from-motion (SFM), which we touched on briefly in Chapter 2. Here we present some examples.

Zisserman et al. [37, 151] propose to reconstruct a 3D scene from point and line matches. They use the Harris interest operator to find a moderate number (around 500) of salient points in each frame of a video sequence. They use cross-correlation to generate match candidates, and a robust estimation method to find the epipolar geometry and correct matches. They verify the matches across three views using the tri-focal tensor, and then extend line matches across three views. Using all the correspondences, they estimate the camera positions for triples of images, and extend the estimate over the entire sequence via overlapping triples. Finally, they refine the 3D points and camera positions using bundle adjustment and perform a Euclidean upgrade. They propose several ways of using the point cloud to create models of the scene for virtual and augmented reality.

Pollefeys et al. [99, 100] develop a theory of metric upgrades and apply it to dense reconstructions of 3D scenes. The key point of the theory itself is that it is reasonable to make limited assumptions about real-world cameras, such as zero-skew pixels, and that by doing so we can go from projective calibration to metric calibration without explicit knowledge of intrinsic parameters. Their practical system works by finding sparse matches, computing a projective calibration, performing the metric upgrade, and then finding dense matches using standard stereo methods. An interesting part of their approach is warping images so that their epipolar lines are horizontal pixel rows (scan lines), an idea similar to that used by Debevec [28].

These methods are limited to modeling single rigid components. When the image sequence contains multiple objects moving independently, it is necessary to segment the im-

age measurements into rigid groups in order to apply structure-from-motion.

### 4.2.3 Affine Motion Segmentation

Several existing algorithms for motion segmentation rely on affine structure-from-motion constraints to find groups of rigidly-moving points in image sequences [12, 24, 45]. Gear [45] points out that the most difficult step in rigid motion segmentation is finding the number of independently moving rigid bodies. He gives a method for estimating this on noisy data by computing the *reduced row-echelon form (RREF)* of the data matrix. The data matrix contains the 2D locations of $n$ points seen in $m$ views, and can be written

$$\mathcal{D} = \begin{bmatrix} \boldsymbol{p}_{11} & \boldsymbol{p}_{12} & \cdots & \boldsymbol{p}_{1n} \\ \boldsymbol{p}_{21} & \boldsymbol{p}_{22} & \cdots & \boldsymbol{p}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{p}_{m1} & \boldsymbol{p}_{m2} & \cdots & \boldsymbol{p}_{mn} \end{bmatrix}.$$

Its factorization, written $\mathcal{D} = \mathcal{M}\mathcal{N}$, has the following form:

$$\mathcal{D} = \begin{bmatrix} \mathcal{M}_{11} & \mathcal{M}_{12} & \cdots & \mathcal{M}_{1k} \\ \mathcal{M}_{21} & \mathcal{M}_{22} & \cdots & \mathcal{M}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_{m1} & \mathcal{M}_{m2} & \cdots & \mathcal{M}_{mk} \end{bmatrix} \begin{bmatrix} \mathcal{N}_1 & 0 & \cdots & 0 \\ 0 & \mathcal{N}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{N}_k \end{bmatrix}, \tag{4.1}$$

where there are $m$ camera matrices $\mathcal{M}_{ij}$ for each of $k$ independently moving objects, and each object contains a (generally different) number of 3D homogeneous points appearing in a block $\mathcal{N}_j$. From the factorization, we observe that the rank of $\mathcal{D}$ must be no greater than $4k$.

The RREF of a matrix $\mathcal{U}$ is a matrix $\mathcal{U}'$ whose rows are linear combinations of the rows of $\mathcal{U}$ and that satisfies the following conditions: (1) all rows consisting entirely of zeros are at its bottom; (2) the first nonzero entry in each row is a 1; (3) the leading 1 in each row is to the right of all leading 1s in rows above it; and (4) each leading 1 is the only nonzero entry in its column. The columns in which leading 1's occur are called *base columns*. The base columns form a basis for all the other columns in $\mathcal{U}'$, and thus determine its rank $r$.

The RREF $\mathcal{D}'$ of the data matrix $\mathcal{D}$ should provide the value for the unknown number of objects $k$ and also which points belong to which objects. If a column of $\mathcal{D}'$ contains a non-zero entry, then the leading 1 on the same row as the entry is part of the basis for the subspace in which the column resides. Ideally, there are $k$ such subspaces, and a simple

graph construction should reveal which columns belong to them. In practice, noisy data and numerical errors cause $\mathcal{D}$ to have full rank and all the columns in the graph construction to be connected. Gear assumes that the measurements contain Gaussian noise, and provides a method drawing on numerical and probabilistic techniques to estimate the actual partition of the points into rigid objects.

Costiera and Kanade [4] proposes a different method, based on a factorization of the data matrix $\mathcal{D}$. Using the same setup as (4.1), we have $\mathcal{D} = \mathcal{M}\mathcal{N}$, and $\mathcal{D}$ has (at most) rank $4k$. Now the rows of the matrix $\mathcal{N}$ form a basis for the $4k$-dimensional subspace spanned by the rows of the matrix $\mathcal{D}$. The operator that maps any vector onto its orthogonal projection into the space spanned by the columns of a matrix $\mathcal{A}$ can be represented by the matrix $\mathcal{Z} \stackrel{\text{def}}{=} \mathcal{A}(\mathcal{A}^T\mathcal{A})^{-1}\mathcal{A}^T$.

Costiera and Kanade call $\mathcal{Z}$ the *shape interaction matrix*. It will be block diagonal if the columns of $\mathcal{D}$ are ordered correctly. In general this is not the case. The values in $\mathcal{Z}$ (modulo column and row swaps) are independent of the order of the columns of $\mathcal{D}$. Thus, recovering the correct point ordering (and the corresponding segmentation into objects) amounts to finding the row and column swaps of the matrix $\mathcal{Z}$ that reduces it to block-diagonal form. Costiera and Kanade have proposed several methods for finding the correct swaps in the presence of noise. One possibility is to minimize the sum of the squares of the off-diagonal block entries over all rows and column permutations.

Several other approaches to the affine motion segmentation problem exist. They are based on alternate ways of applying the rank constraint on the data. The affine-subspace method uses the observation that the projected points of an object can be described by three basis points in each image and a 3D coordinate vector for each point on the object [122, 144]. Generalized Principal Component Analysis (GPCA) casts the problem of determining the number of subspaces and the basis for each subspace in terms of polynomial factorization [139].

These methods tend not to work well on shots from movies, which often contain degenerate structure or motion. Some scenes, such as urban streets, may contain significant global perspective effects, limiting the applicability of affine methods. The approach we take is to apply projective constraints directly in the motion segmentation process, which resolves the problem of perspective effects but is still vulnerable to degeneracies.

Fitzgibbon and Zisserman [39] discuss metric upgrades on scenes that contain multiple rigid objects. They describe a simple segmentation algorithm based on robust estimation which they attribute to Torr [131]. The procedure iterates between two key steps: 1) Use RANSAC [36] to select the dominant motion. 2) Subtract all data in the dominant motion, leaving only data in other rigid components. The procedure repeats until the number of

data is too small to reliably estimate the multi-view geometry of the image sequence.

Sivic and Zisserman [122] take another approach to describing whole shots which is similar to the one proposed in this thesis, in that they track affine-covariant patches across an image sequence and segment them into motion groups. Specifically, they process each frame of a shot with Harris and Matas detectors, further processing the results to obtain non-oriented affine regions. These are linked across frames to form tracks. A combination of several local motion constraints and an affine-subspace model produce a motion segmentation for the tracks. By limiting the number of frames covered by the motion constraints, they are able to handle small amounts of non-rigidity in the objects. Finally, they match between motion components in the same shot to detect if any belong to the same object undergoing an occlusion event.

## 4.3 Modeling

A shot is a strict linear sequence of images, rather than the more general graph of image relationships in the case of photos (Chapter 3). A surface patch will appear in some frame of the sequence, continue to appear in each subsequent frame, and then disappear in some later frame. A surface patch tracked through a contiguous sequence of frames is called a "track". Tracks are the basic feature for scene reconstruction. Motion segmentation seeks subsets of the tracks in a shot that are consistent with a single rigid object moving in 3D. Each rigid component forms a 3D model via SFM on the image measurements contained in the tracks.

### 4.3.1 Tracking

Because successive frames of a video are close in time and space, the feature matching problem simplifies to point tracking, and we use a standard solution: the Kanade-Lucas-Tomasi (KLT) tracker. Given two images taken close to each other in space and time, and given a point in one image, KLT finds its match in the other image. KLT iteratively searches for the location of the new point by minimizing the pixel differences in a fixed window around the point in the "old" image and the point in the "new" image. The actual calculations are similar to the ones given in Section 3.2.1 for patch refinement, but the solution is linear and only for location.

KLT tracks points, but we need to track affine-covariant patches, so we have augmented Birchfield's implementation [11] of the KLT tracker as follows: For each new frame $i$, we find points in the image that aren't currently being tracked and determine their patch

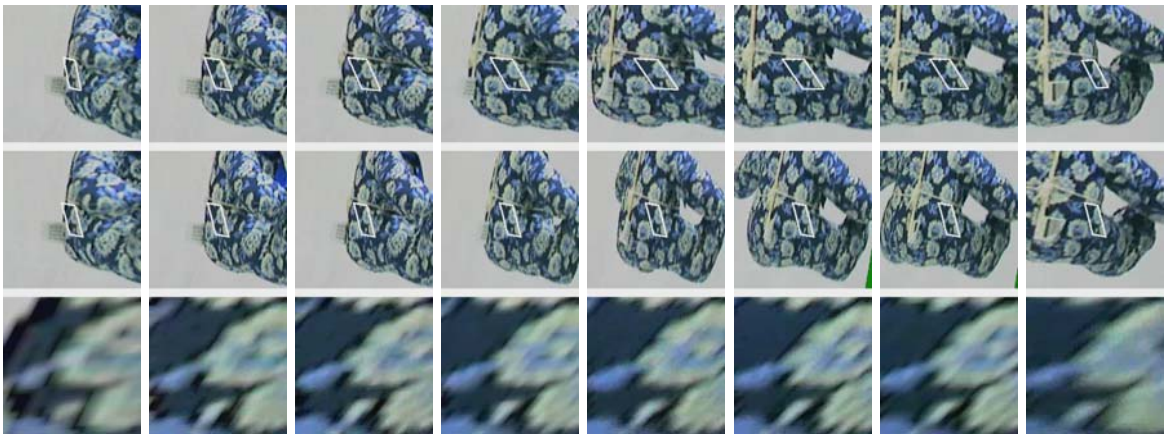


Figure 4.1: A tracked patch. Top row: the patch marked in the original video. Middle row: the patch stabilized such that it maintains a constant shape and the surrounding image deforms. Bottom row: the rectified patch. This figure shows every eighth frame.

parameters using the affine-adaptation process described in Chapter 2, providing an initial value for the matrix $\mathcal{S}_{ij}$ associated with each patch $j$. For all patches that are currently being tracked (i.e., that exist in frame $i-1$), we use the KLT tracker to update the location of the patch center in frame $i$, and then use non-linear least squares to refine the parameters of the patch, maximizing the normalized correlation between the patch in frame $i$ and the same patch in the frame where it first appeared. In addition to the criteria that KLT itself uses to stop tracking a point, we also check whether the ratio of the dimensions of the patch exceed some threshold (typically 6), and whether the correlation with the initial patch falls below some threshold (typically 0.8). In a later step we trim the tracks using stricter thresholds in order to minimize geometric error, but during the tracking phase it is important to track each patch as long as possible.

It is possible for a patch to disappear and reappear in the sequence, such as when an object passes temporarily behind another object. We treat such a case as two different tracks. They can of course be unified by an internal matching procedure, such as Sivic's "track repair" [122].

It takes an average of 30 seconds to process one frame of video. In practice, our tracking technique gives excellent results, yielding very robust tracking results as well as sub-pixel localization, which is crucial for the reliability of the multi-view constraints used in segmentation and modeling.

### 4.3.2 Motion Segmentation

Our approach to motion segmentation uses rigid 3D modeling and rigid consistency testing as atomic components. Below we will give some details on how to implement these in the context of video, but for now we will take them as "black boxes" and discuss how the motion segmentation strategy itself works.

**Input:**
• A set of tracks $T$.
• A threshold $o$ on the minimum frames two tracks must share to "overlap". This controls how long two tracks must move together to give high confidence that they are rigidly connected.
• A threshold $t$ on reprojection error. This determines if a track is consistent with a model.
**Output:** A set of rigid groups and their associated 3D models.

**repeat**
• Find the frame $f$ with the largest number of concurrent tracks in $T$. A track must appear at least in frames $[f, f + o)$ to qualify. Call the set of overlapping tracks $O$.
• Use RANSAC to find the largest subset of tracks in $O$ that are rigidly consistent: For each random pair sampled from $O$, form a 3D model and then select all other tracks from $O$ with reprojection error below $t$ to form a consensus set. Keep the largest consensus set and call it $C$.
**repeat**
• Form a model from $C$.
• Replace $C$ with all tracks in $T$ with reprojection error below $t$.
**until** $C$ stops growing.
**if** $C$ contains a sufficient number of tracks **then**
• Add $C$ and its model to the output.
• $T \leftarrow T \setminus C$
**end if**
**until** another $C$ with enough tracks cannot be formed.

**Algorithm 6:** Motion Segmentation.

Algorithm 6 summarizes our method, which follows the approach suggested by Fitzgibbon [39]. It first locates the frame in the video that has the largest number of concurrent tracks. This provides the richest set of data for detecting the dominant motion. Note that at all stages of the processing, tracks must overlap by some minimum number of frames (typically 6) in order to determine that they move together rigidly. Algorithm 6 selects the dominant motion among the concurrent tracks using RANSAC, and then grows the associated rigid component by adding consistent tracks from anywhere in the shot until the set of tracks consistent with the model no longer changes between iterations (or it cycles among a small number of values). Finally, it subtracts the rigid component from the set of free

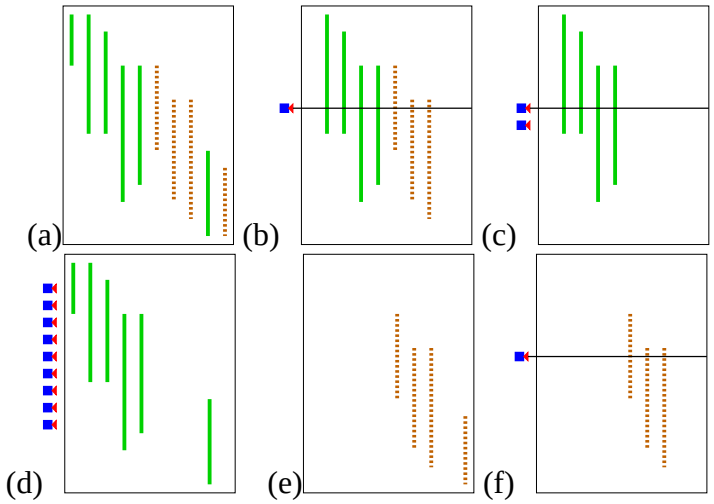



Figure 4.2: Motion segmentation of two rigid components. The vertical dimension is time and the horizontal dimension is patches. (a) tracks, (b) thickest overlap in video, (c) largest component in overlap, (d) grown model, (e) after subtraction, (f) start of next iteration.

tracks and repeats from the RANSAC step. This process stops when it is no longer able to collect a sufficiently large set of tracks.

Figure 4.2 illustrates the process of segmenting tracks. Each bar in Fig. 4.2(a) is a track associated with a surface patch, with the vertical dimension representing time (i.e., frames). In this toy example there are two rigid components, indicated by different colors. In the first step (Fig. 4.2(b)) the algorithm seeks the frame in the video where the largest number of tracks are simultaneously visible for two consecutive frames. (For the sake of illustration this overlap threshold is minimal.) The single camera and horizontal line indicate the selected frame. Only the set of tracks in $O$ are shown. Using RANSAC, the algorithm discovers one of the rigid components (Fig. 4.2(c)). The two cameras indicate the recovered poses in the 3D model associated with the component. The algorithm then grows the component as much as possible (Fig. 4.2(d)), and removes it from the set of unsegmented tracks (Fig. 4.2(e)). In Fig. 4.2(f) the algorithm starts a new cycle, looking for the place in the video that contains the most concurrent tracks.

The algorithm builds a 3D model for each rigid component of the scene as an integral part of its processing. There is no separate 3D modeling stage after motion segmentation. Also note that the algorithm gives us a principled approach to motion segmentation that does not depend on determining the rank of the data matrix.

### 4.3.3 Handling Missing Data

Here we provide specifics on how modeling works in the context of video. The approach to modeling described in this thesis depends on a set of image measurements that are all associated with a single rigid object. The motion segmentation method described above attempts to select subsets of the tracks that meet this requirement.

The selected tracks populate a patch-view matrix and we must, as in Section 3.2.2, solve the missing data problem. However, in the case of video the problem is simpler for two reasons. First, we use an alternate method for assembling the model called *bilinear merging*, presented below, which removes the need for all but one large initial block. Second, the problem of finding maximal dense blocks is more constrained in the case of tracks (contiguous sets of image measurements) than the case of patches in a sparse set of images.

Specifically, a patch-view matrix with contiguous tracks is equivalent to an interval graph. In general, an interval graph is one in which each vertex represents a contiguous range (such as intervals on the real number line or circular arcs in an arc graph) and each edge represents an overlap between two ranges. In our case, each vertex represents the unbroken sequence of views in which a surface patch appears. The edges are sequences of views where two given surface patches are both visible. A clique (that is, a fully connected subset of the vertices) in the graph is equivalent to a dense block, so finding maximal cliques is equivalent to finding maximal dense blocks. Maximal cliques can be found in interval graphs in polylogarithmic time, rather than NP time as required for the general case [51].

Algorithm 7 gives all the maximal cliques/blocks with at least $N_V$ views. It is inspired by [51], though it solves a slightly different problem. It enumerates all blocks that could be maximal and selects those that actually are. It ignores most trivial blocks that are simply a sub-block of some other block. Assume there are $m$ views and $n$ tracks, and for simplicity that $n \geq m$. There are $m$ possible views where some block may begin, and $m$ possible views where some block may end. A block is determined entirely by its start and end positions: exactly those tracks that are present at both endpoints will be in the block. The blocks found by the algorithm are maximal in the sense that none of them are subsets of some other larger block. To see this, note that a block so constructed cannot grow by a view without either becoming sparse or giving up some number of tracks. Similarly, a block cannot add a track without either becoming sparse or giving up one or more views.

It takes $O(n)$ time to find the intersection between two sets of tracks. The overall run time is therefore $O(nm^2)$. We typically enumerate all maximal dense blocks and choose the largest one, meaning that it takes the full $O(nm^2)$ time to find the largest dense block.

**Input:** For each track, the indices of the first and last views in which it appears.
A lower limit $N_V$ on the number of views in a block, $N_V \geq 2$.
A lower limit $N_P$ on the number of tracks in a block, $N_P \geq 2$.
**Output:** A set of dense blocks of views $\times$ tracks.

Shorten each track by $N_V - 1$. That is, for each tracked patch, subtract $N_V - 1$ from the index of its last view. Only retain tracks with positive length.
Always handle views in sequential order...
**for all** views $V_i$ where some track starts **do**
  **for all** views $V_j$ where some track ends, $j \geq i$ **do**
    $B \leftarrow$tracks$(V_i) \cap$tracks$(V_j)$.
    **if** at least one track in $B$ starts at $V_i$ and at least one track in $B$ ends at $V_j$ **then**
      Create a block consisting of tracks in $B$ and views from $V_i$ to $V_j$ inclusive.
    **end if**
  **end for**
**end for**
Lengthen each block by $N_V - 1$ views.

**Algorithm 7:** Contiguous Blocks.

We are aware that an $O(n \log n)$ algorithm exists for enumerating maximal cliques of an interval graph [80], but this is little need to implement the best possible algorithm since this part of the process takes a trivial amount of time in any case.

After choosing the largest dense block, we form a reconstruction from it. The resulting model provides a starting point for bilinear merging, which in turn adds all the other tracks to the model.

### 4.3.4 Bilinear Merging

Algorithm 6 often needs to add tracks to an existing model, because during the region growing part of the process each new set of tracks tends to be a superset of the previous one. We could rebuild the entire model, including both the new and pre-existing tracks, generating dense blocks and registering them as described in Chapter 3. Such an approach would be prohibitively expensive. On the other hand, the method presented below builds models incrementally using a modification of the bilinear bundle adjustment routine, and is thus more vulnerable to getting stuck in a poor solution than the method that registers blocks.

Bilinear refinement (Algorithm 2 presented in Section 2.2.4) provides a practical tool for integrating new data into an established model. It works on sparse data, so new data is not required to cover exactly the established views or patches in the model. And it allows us to hold the cameras constant while estimating the patches (and vice-versa). This is the

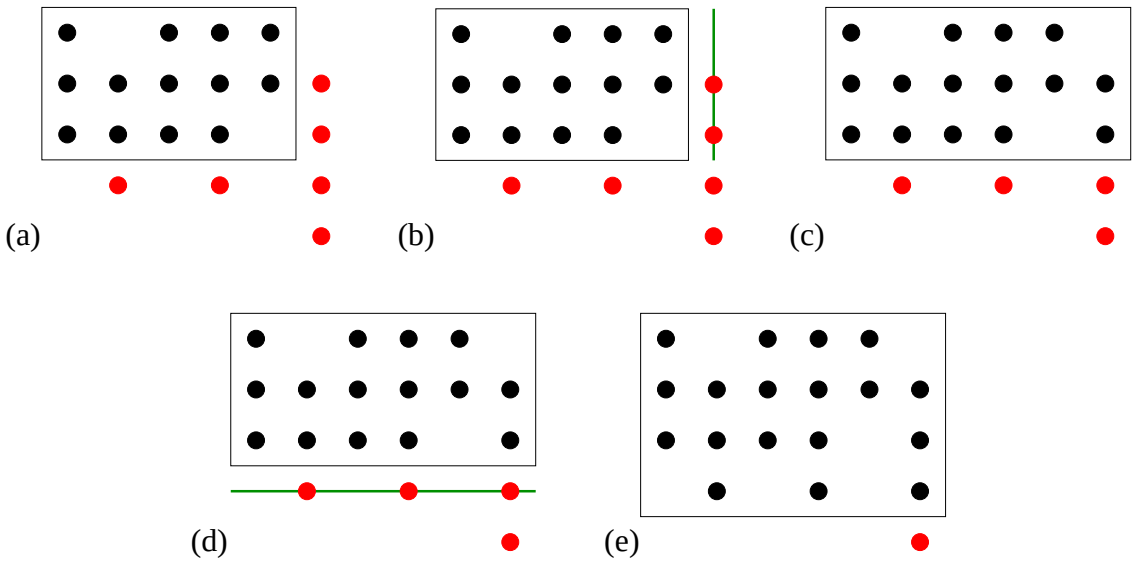



Figure 4.3: Merging tracks into an existing model, illustrated in terms of the patch-view matrix: (a) the initial model, (b) estimating a patch from a new track, (c) the model after adding the patch, (d) estimating a new camera, (e) the model after adding the new camera.

key that allows us to add a new track and its associated patch to the model.

Figure 4.3 illustrates the process of adding a track to a model. Dots represent individual patches measured in individual images. Columns represent tracks and their associated 3D patches, and rows represent images and their associated cameras. The vertical and horizontal lines indicate "slots" where image measurements would be used for a particular estimate if they existed. (Of course, only measurements that actually exist are used.) In this scenario, some tracks that are already part of the model contain data in views that are not part of the model. This is actually the common case, and is due to the fact that only cameras with a sufficient number of supporting data are including the model. Figure 4.3(b) shows a patch being estimated from two of the three known cameras in the model and two of the four new image measurements in the track. Once the new patch is estimated, there is now enough data to support estimating another camera. Figure 4.3(d) shows this camera being estimated from three of the six known patches in the model and three image measurements.

In practice, when adding a number of tracks to the model, we choose the track or camera to add next based on which one has the largest overlap with the current model. This tends to minimize the amount of estimation error introduced into the model. At the same time, we check the reprojection error of the estimate to guard against adding outliers. Periodically, we perform a few (typically 4) iterations of bilinear refinement on the current model to allow the new data to propagate to the established cameras and 3D patches.

A potential difficulty with the approach sketched above is that the point arrangements

or camera motions contained in the overlap may contain degeneracies. For example, the patches marked by the bar in Fig. 4.3(d) may contain (nearly) coplanar points, preventing the reliable estimation of the camera matrix associated with that row. Again, the strategy of adding the camera or patch with the largest overlap will tend to minimize the chance of this happening.

### 4.3.5 Results

It is difficult to get a sense of the processing or output of our proposed system without seeing full motion video. See Appendix C for a CD of video results in MPEG format. We also provide examples on our web site: `http://www-cvr.ai.uiuc.edu/ponce_grp/research/3d`. Here, we present select snapshots.

The number of components found by the segmentation program depends on the choice of parameters for the modeling process (i.e., Algorithm 6). We generally tune the program to oversegment slightly. This does not necessarily hamper applications such as shot matching since multiple models can in principle be matched independently.

Figure 4.4 show the results of a laboratory experiment using videos of stuffed animals. The first row shows a segmentation experiment where the head of a bear is moved by hand independently from its body. The head is found as one segment, and the body as another. The second row of the figure shows a segmentation experiment using the bear and a dog rotating independently, but with similar speeds and axes of rotation. Representative frames of the video are shown in the figure, along with the corresponding patches and reprojections of the estimated models, surrounded by a black frame. The dog in the bear-dog video is difficult because the features on the narrow ends are short-lived and do not provide a strong connection between the two sides. The segmentation program finds two or even three components for the dog, depending on parameter settings. Figure 4.4 shows a test with exactly two components, one for the dog and one for the bear. However, the dog model is not fully self-consistent, in that reappearing patches are not well registered. The reconstruction tends to be better when the dog is broken up. The remainder of the figure shows the bear model constructed from the bear-dog video, along with the recovered cameras.

Figure 4.5 shows results of segmenting and modeling shots from the movies “Run Lola Run” and “Groundhog Day”. The first row of the figure shows a corner scene from “Run Lola Run”. The two rigid components are the car and the background. The program actually finds multiple segments for the background (only one is shown). The background is challenging, both because the shot contains a rapid pan and because there is a large number

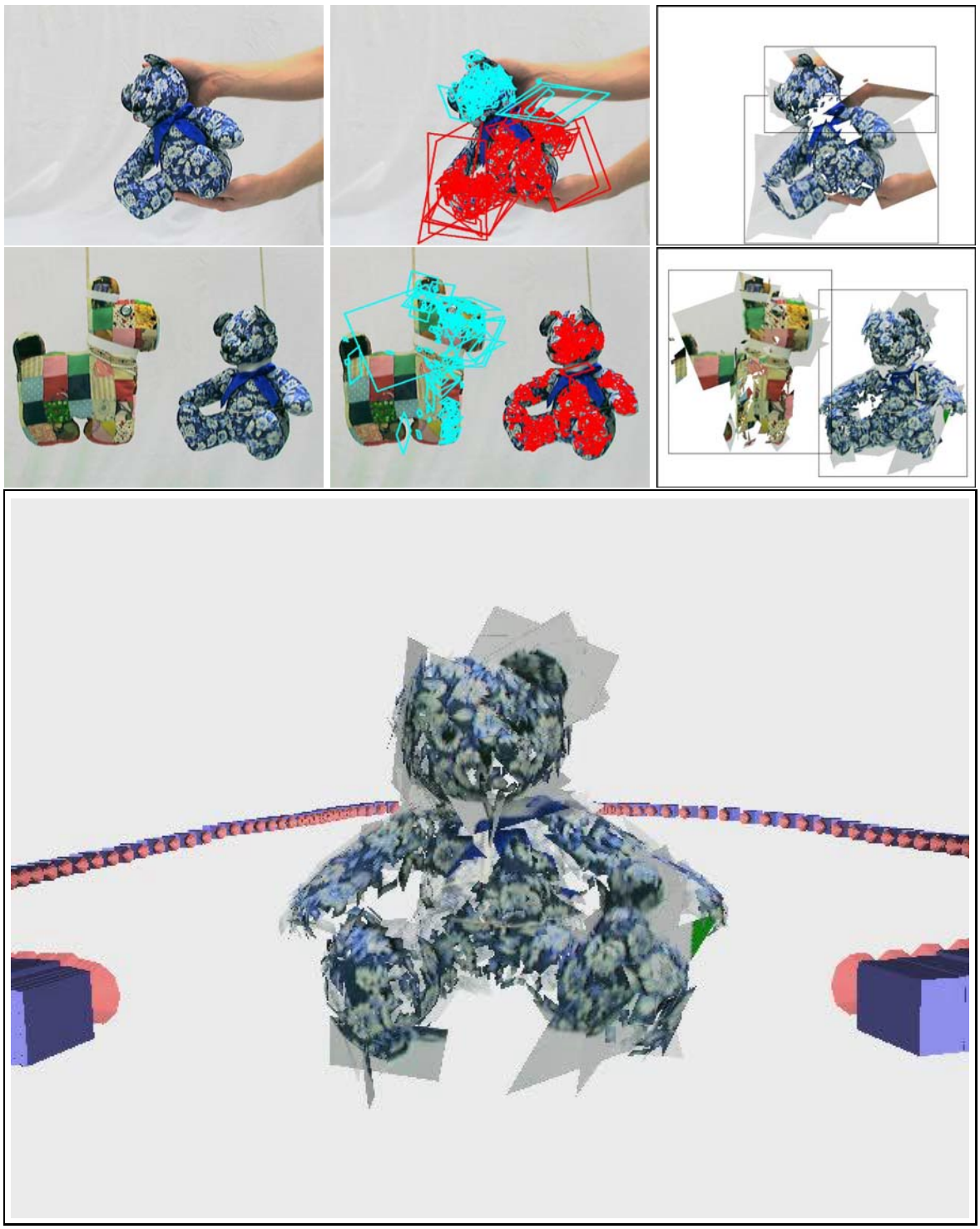


Figure 4.4: Segmentation and modeling of lab videos.

of frames where all the points are essentially co-planar. Films tend to be difficult in general because they frequently use nearly degenerate motions such as pans in place or translating along a straight path, or even no motion at all except in the non-rigid components of the scene. The second row of Fig. 4.5 shows a scene from "Groundhog Day". The rigid components are the van and the background. Later, another vehicle turns off the highway and is also found as a component. The last row of the figure is a reprojection of the 3D model of the van. Note that the viewpoint of the reprojection is significantly different than any in the original scene.

## 4.4 Recognition

Videos are commonly segmented into shots for the purposes of annotation or indexing [65, 95]. A video indexing system searches for shots that contain the same object or scene as a query [3, 112]. Alternatively, one may wish to cluster similar shots in order to analyze the larger structure of the video. Our goal is to demonstrate the ability to measure the similarity between shots. This could form the basis of either a retrieval or a clustering system.

For the purposes of this experiment, it is useful to have video with multiple repeats of the same scene. "Groundhog Day" and "Run Lola Run" are popular candidates. The movie "Run Lola Run" contains three repetitions of roughly the same plot sequence, with slight variations. Figure 4.6 illustrates this. Note that we use the shot segmentation provided by the VIBES project [138] for "Run Lola Run". We determined the shots for "Groundhog Day" by hand, with some help from Josef Sivic.

There are several possible approaches for using 3D models to match two video shots. Three rather direct applications of the machinery in this thesis are:

- Form a model from one shot and match it to a key frame in the other shot.
- Form a model from one shot and match it against the tracks in the other shot.
- Form models of both shots and directly compare them in 3D.

We have experimented briefly with the second approach, but currently use the third. The advantage of the third one over the second is that the data to compare are more compact (reducing the required computation) and allow the strongest form of 3D constraint. The advantage of either approach over the first is that they make use of the motion in the shot.

In this context, the recognition problem is as follows: A "query" is a single rigid component selected from a database of models. Given a query component, return all shots that

Figure 4.5: Segmentation and modeling of shots from “Run Lola Run” and “Groundhog Day”.

Figure 4.6: Frames from two different scenes in "Run Lola Run". Each scene appears three times in the movie.

contain a closely matching component. The scenario is a user operating a video retrieval system, querying for shots that contain the given object. In our naive implementation, the system compares the query object to each component in a candidate shot, and returns all shots that contain some component that matches with sufficient confidence.

We apply Algorithm 3 between the query model and a given component, called the "test" model in the sequel. The following sections give the details of the algorithm in the context of matching 3D models.

### 4.4.1 Appearance-Based Selection of Potential Matches

We describe the patches in each model using SIFT and color histograms, in a manner similar to recognition in photographs (Section 3.3.1). Color provides an initial filter on potential matches, which we then compare using the SIFT descriptor. For a given patch in the query model, we select the top $K$ patches in the test model.

At this point in previous applications of Algorithm 3, we typically filter match candidates with a neighborhood constraint. Due to the (relatively) low cost of direct 3D matching, this is not really necessary. However, a neighborhood constraint is clearly possible. Each 3D patch can establish a local affine frame of reference, and the centers of nearby patches can be projected onto the plane supporting the primary patch. A similar neighborhood can be established in the other model to verify nearby matches.

### 4.4.2 Robust Estimation

The set of matches between the two 3D models determines a registering transformation (or "registration"). The exact form of the transformation (similarity, affine or projective) depends on the form the respective models. A "Euclidean" model is one whose points are separated from the true 3D points of the original object in the world by a similarity transformation. Likewise, "affine" and "projective" models are separated from the true shape of the object respectively by affine and projective transformations. Two models should be registered by the most general transformation separating either one of them from the true shape of the object. However, our experiments show that an affine registration provides better results, even when one or both models are projective. Affine registration is more robust against noise due to differences in the recovered patches between the two models, and against degeneracies (e.g., coplanar points). Lowe makes a similar observation in the context of aligning 2D models [72].

The error measure for a set of matches is the root mean squared distance between the respective points in the two models after applying the estimated registering transformation between them. The distance measure between a given pair of matched points is the Frobenius norm of their difference divided by the characteristic scale of the 3D patch in the query model. The idea is that larger scale patches have less certain localization, and so should have a more relaxed distance measure.

### 4.4.3 Geometry-Based Addition of Matches

Given the registering transformation between the two models, we project the patches from the test model into the query model. For each patch in the query model, we then select the $K$ nearest patches in the test model.

### 4.4.4 Object Detection

We use three measures of the quality of the registration between the two components. The first measure is the repeat rate [116], which is defined as $M/\min(A, B)$, where $M$ is the number of trusted matches, and $A$ and $B$ are the number of 3D patches in the respective components. This number can range from 0 to 1, where 0 means no matches and 1 means everything matches.

The second and third measures are based on the amount of deformation encoded in the registering transformation $T$ between the two models. ($T$ is estimated by the matching algorithm.) The second measure is the ratio of the largest amount of scaling over the

smallest amount of scaling. Specifically, if $\lambda_{min}$ and $\lambda_{max}$ are the smallest and largest eigenvalues of $T_{3\times3}T_{3\times3}^T$, where $T_{3\times3}$ is the upper-left $3 \times 3$ sub-matrix of $T$, then the second measure is $\sqrt{\lambda_{max}/\lambda_{min}}$. Essentially, the matrix $T_{3\times3}T_{3\times3}^T$ describes an ellipsoid in 3D, and the eigenvalues are related to the scales of its principal axes. Ideally, the ratio will be close to 1, though many models require some amount of non-uniform scaling to register properly. Mainly, this measure tests for distortion in the transformation due to matching completely unrelated patches.

The third measure is the amount of skew. Specifically, if $R_1^T$, $R_2^T$, and $R_3^T$ are the three row vectors of $T_{3\times3}$, then the measure is

$$\max\left(\frac{|R_1^T R_2|}{\|R_1\|\|R_2\|}, \frac{|R_2^T R_3|}{\|R_2\|\|R_3\|}, \frac{|R_3^T R_1|}{\|R_3\|\|R_1\|}\right).$$

A large amount of skew again indicates distortion due to matching unrelated patches.

We test each measure against a threshold and combine them with a simple AND test. That is, if all three measures pass their thresholds, then the object is recognized.

### 4.4.5 Results

The database for the shot matching experiment contains 3D models collected from various sources: lab videos, “Groundhog Day”, “Run Lola Run”, and still images. The database contains one still model (the bear) and 27 shots consisting of 78 components total. Figure 4.7 and 4.8 show a gallery shots in the database.

We selected 10 components from the database to act as query objects, and tested each query object against all the other objects in the database. Figure 4.9 gives the overall performance of the shot matching system. As in the still image case, there are several parameters that control recognition. We systematically sample the parameter space and report the maximum recognition rate for each number of false positives.

Figure 4.10 shows some successful matches. The background is a grayed-out frame from the recognized shot. The foreground contains the patches, in color, from the query model. These results are best viewed in motion, and sample videos appear in Appendix C and on our web site.

## 4.5 Discussion

*The contributions of this part of the thesis are:*

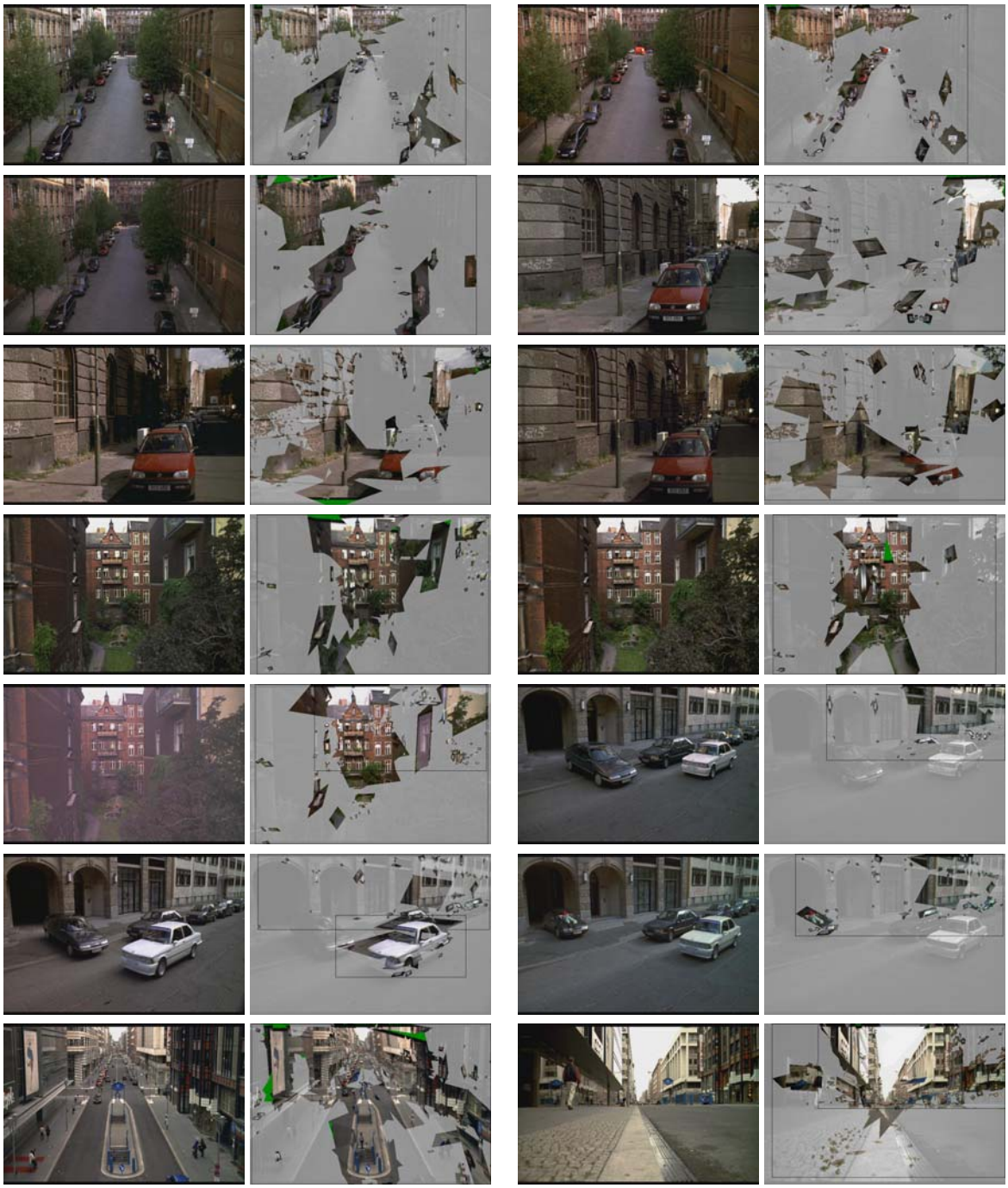


Figure 4.7: Gallery of shot models. The left image is the raw frame and the right image is marked up with reprojected patches.

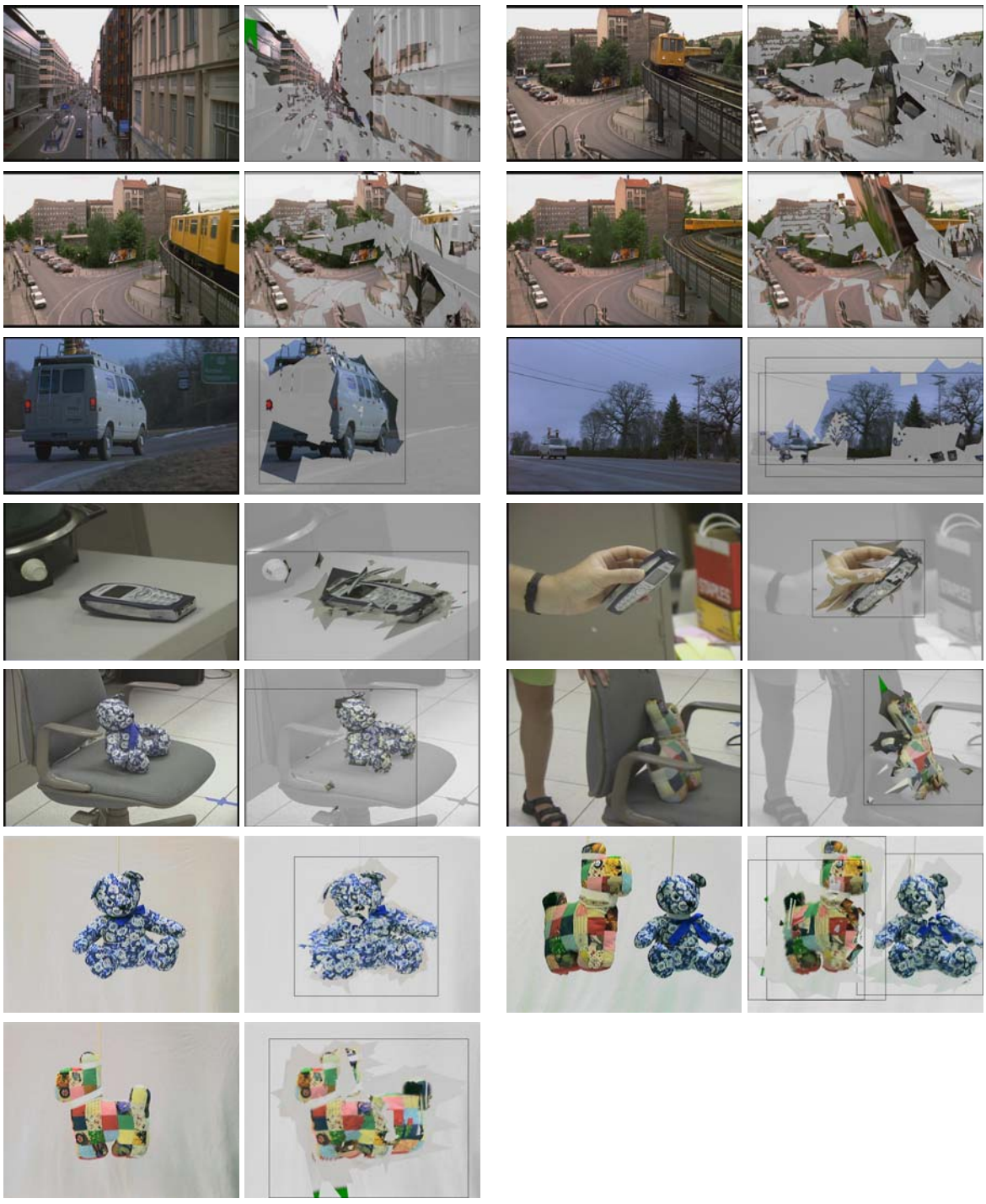


Figure 4.8: Gallery of shot models. The left image is the raw frame and the right image is marked up with reprojected patches.

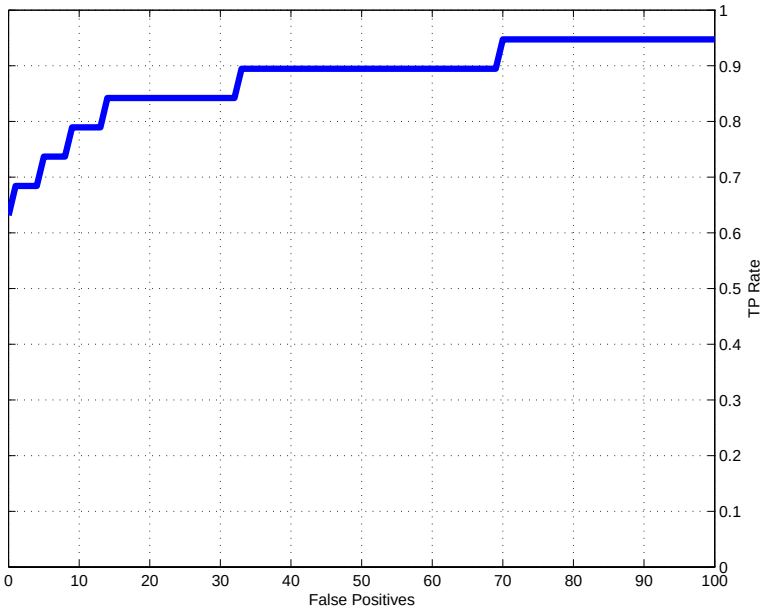



Figure 4.9: Recognition rate versus false positives.

Figure 4.10: Some correctly matched shots. The left image is the original frame of the test shot. The right image shows the query model reprojected into the test video.

- *A new approach for modeling rigid objects from a video sequence and articulated object modeling.*

- *A video description and shot matching method that makes full use of structure from motion, recovering both 3D objects and their poses.*

With few exceptions [64], video retrieval researchers have made very little use of structure from motion. Most video description methods only recover the motion of 2D regions, and there are none that we know of that use 3D structure for shot matching. One reason for this may be that arbitrary video tends not to be "friendly" to structure from motion. Sometimes cameras are stationary, and when they move the motion is nearly degenerate. The system presented in this chapter gracefully degrades to planar models in the case of degenerate motion, and many of the shots tested are nearly planar. However, even in the difficult cases there is sufficient structure to do true 3D matching. Furthermore, when rich structure is available, it becomes a powerful matching constraint.

The tests reported here are a proof of concept on a small set of data. The goal is to show that 3D constraints are useful for video analysis and shot matching. There is room for improvement in both the speed and accuracy of the implementation. We believe that it would be useful to develop a standard data set and scenario for testing video retrieval methods, and to compare those methods quantitatively. Such a data set probably should contain a wide range of material, portraying varying types of activities (e.g., newscasts, sports, action scenes, etc.) and scenery (e.g., urban, country, indoors, public areas, etc.).

The proposed approach is limited to rigid objects. However, many interesting objects in videos are non-rigid, the prime example being humans. Even a piecewise rigid model is not sufficient to model humans, because they tend to be clothed and cloth tends to move non-rigidly. Furthermore, humans tend to appear at a scale level that yields a small total number of patches, while our method requires a good number of patches to detect an underlying object. Future work will focus on extending our method to handle these cases.

Our implementation of the locally-affine construction has some weaknesses. The inverse projection matrix $\mathcal{N}$ associated with a given 3D patch has an affine form. Specifically, all the homogeneous 3D coordinates in the patch have the same scale. If this scale was allowed to vary (two additional degrees of freedom), we may be better able to model some scenes where the objects are relatively close to the camera and there is significant change in the angle between the surface normal and the viewing direction during the course of a scene. We have not yet tested this generalization to see how much difference it would make in practice. The drawback of such an approach is that the extra degrees of freedom may lead to overfitting and more sensitivity to noise.

Another possibility is to describe each image measurement (that is, the inverse rectification $\mathcal{S}$ from the normalized form to the image patch) as a full homography rather than an affine transformation. Doing so would amount to abandoning the locally-affine construction in favor of a fully general perspective one. Generally, perspective deformations are not measurable within a single patch [135]. However, a patch changes shape over the course of a shot, and the two extra degrees of freedom could reduce the error in the measurements by a beneficial amount. Again, the extra freedom could also allow more error, so this should be evaluated carefully.

# Chapter 5

# Discussion

## 5.1 Thesis Contributions

This thesis makes the following contributions:

- A new framework for object recognition where object models consist of a collection of (small) planar patches, their invariants, *and* a description of their 3D spatial structure.
- An implemented method for automatically acquiring 3D models of rigid objects from a small set of unregistered photographs and recognizing them in cluttered photographs taken from unconstrained viewpoints
- An implemented method for automatically acquiring 3D models of articulated objects from image sequences, and for measuring shot similarity based on these models.

The approach presented here fully exploits the 3D constraints implied by the multi-view geometry. Most other image matching methods take advantage of some geometric constraints, but those constraints tend to be "weak" in the sense that they are heuristics that only approximate the expected effects in 2D of the multi-view geometry. If the exact shape is known (specifically, if the object is rigid and seen from a sufficient range of viewpoints), then it should be exploited. This approach does not address variation in object shape. On the other hand, a method based on "weak" constraints indirectly tolerates some global deformation (see, for example, [35]).

The approach of representing object structure as planar patches in 3D synthesizes a number of other areas of research: shape-from-texture, wide-baseline matching, structure-from-motion, appearance-based and geometry-based recognition. Such a synthesis is valuable since it promotes some transfer between these lines of research.

The need for overlap between training images is a limitation of this approach. Methods which represent an object as a collection of independent views do not require such dense coverage. A possible compromise, currently under research in our group, is to use stereo pairs to get some of the benefit of 3D structure, but to avoid doing a full reconstruction of the object.

The 3D patches described in this thesis indicate the tangent planes of a smooth surface at their respective locations. Tangent planes capture an important part of the structure of an object. However, they do not describe the object as a solid or fully capture its contour. Similarly, local planar images don't fully capture an object's texture. For example, there may be fine structure such as sand or hair that could be described as 3D texture existing very close to the plane. It would be interesting to study a more complete description on an object which combines these elements.

The fact that 2D patches sometime cross the limb of an object in an image can make them unreliable sources of both shape and texture information. On the other hand, at a large scale pieces of the contour of an object do attract point detectors. This could be used to describe the shape of an object, specifically in terms of curved sections in a spatial arrangement.

## 5.2 Future Work

The approach proposed in this thesis is based entirely on rigid modeling. Despite its usefulness in these tasks, rigid modeling does not tell the whole story of vision. We hope that this method will add one small part to the much larger task of solving the grand challenge "problem of vision". To move forward from here, we need to learn how to effectively model flexible objects. One possibility is to conceive of objects as a graph: the vertices are patches and the the edges are 3D distances between the patches. The edges have varying degrees of rigidity. Perhaps we could represent them as distance distributions. This is not the same as an elastic bunch graph [145], which is planar and has fixed relationships between points.

Perhaps most importantly, the rigid modeling method given here cannot handle class-level recognition. To gain some generalization, we may use more general and forgiving descriptors for the patches, and allow more flexibility in matching the 3D structure. However, it seems unlikely that loosening thresholds alone would be sufficient to achieve class-level recognition. This is an exciting opportunity to discover new insights.

# Appendix A

# Inverse Projection Matrices

Here we derive the form of the inverse projection matrix $\mathcal{N}$ introduced in Section 2.2.2. Let $\mathbf{\Pi}$ be the coordinate vector of the plane supporting a given surface patch, and let $\mathcal{M}$ be the $2 \times 4$ affine projection matrix from the scene into its rectified patch. For any point $\boldsymbol{P}$ in the plane, we have $\mathbf{\Pi}^T \boldsymbol{P} = 0$, and projection $\boldsymbol{p} = \mathcal{M}\boldsymbol{P}$. These two equations determine the homogeneous coordinate vector $\boldsymbol{P}$ up to scale. To completely determine it, we can impose that its fourth coordinate be 1, and the corresponding equations become

$$\mathcal{M}_{\mathbf{\Pi}} \boldsymbol{P} = \begin{bmatrix} \mathcal{M} \\ \mathbf{\Pi}^T \\ 0\ 0\ 0\ 1 \end{bmatrix} \boldsymbol{P} = \begin{bmatrix} \boldsymbol{p} \\ 0 \\ 1 \end{bmatrix}.$$

$\mathcal{M}_{\mathbf{\Pi}}$ is an affine transformation matrix, and so is its inverse. If

$$\mathcal{M}_{\mathbf{\Pi}}^{-1} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_3 & \boldsymbol{c}_4 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

we can write

$$\boldsymbol{P} = \mathcal{M}_{\mathbf{\Pi}}^{-1} \begin{bmatrix} \boldsymbol{p} \\ 0 \\ 1 \end{bmatrix} = \mathcal{M}_{\mathbf{\Pi}}^{\dagger} \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix}, \text{ where } \mathcal{M}_{\mathbf{\Pi}}^{\dagger} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \boldsymbol{c}_4 \\ 0 & 0 & 1 \end{bmatrix}.$$

The $4 \times 3$ matrix $\mathcal{M}_{\mathbf{\Pi}}^{\dagger}$ is the *inverse projection matrix* [32] associated with the plane $\Pi$. Note that, for any point $\boldsymbol{p}$ in the image plane, the point

$$\boldsymbol{P} = \mathcal{M}_{\mathbf{\Pi}}^{\dagger} \begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix}$$

lies in the plane $\Pi$, thus $\mathbf{\Pi}^T \boldsymbol{P} = 0$. Since this must be true for all points $\boldsymbol{p}$, we must have $\mathbf{\Pi}^T \mathcal{M}^{\dagger}_{\mathbf{\Pi}} = \mathbf{0}^T$.

$\mathcal{N}$ is simply the matrix $\mathcal{M}^{\dagger}_{\mathbf{\Pi}}$ associated with the plane $\Pi$ and projection matrix $\mathcal{M}$. The projection $\mathcal{M}$ maps the center of the surface patch onto the origin of the rectified image plane. It follows that the coordinate vector of the 3D patch center is

$$\begin{bmatrix} \boldsymbol{C} \\ 1 \end{bmatrix} = \mathcal{N} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

or, equivalently, that $\begin{bmatrix} \boldsymbol{C} \\ 1 \end{bmatrix}$ is the third column of the matrix $\mathcal{N}$. Similar reasoning shows that the form of $\mathcal{N}$ is

$$\mathcal{N} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{V} & \boldsymbol{C} \\ 0 & 0 & 1 \end{bmatrix},$$

where $\boldsymbol{H}$ and $\boldsymbol{V}$ are the "horizontal" and "vertical" vectors from $\begin{bmatrix} \boldsymbol{C} \\ 1 \end{bmatrix}$ to the sides of the parallelogram.

# Appendix B

# Patch Refinement

We use the Levenberg-Marquardt (LM) non-linear least squares algorithm to refine the alignment between pairs of image patches. The error function is essentially the differences between the pixel values in the two rectified patches. The parameters associated with one patch remain fixed, while those of the other patch vary until the error function reaches a local minimum.

Here we give the error function and show how to compute its Jacobian analytically. Let $P(\boldsymbol{x})$ be pixel values from the image containing the variable patch, and let $R(\boldsymbol{u})$ be pixel values from the normalized form of the fixed ("reference") patch, where $\boldsymbol{x}$ and $\boldsymbol{u}$ are homogeneous coordinates with scale fixed at 1. Let $\mathcal{S}$ be the inverse rectification matrix associated with the variable patch. The mapping function between the patches is

$$\boldsymbol{x} = \mathcal{S}\boldsymbol{u} = \begin{bmatrix} u_1 S_{11} + u_2 S_{12} + S_{13} \\ u_1 S_{21} + u_2 S_{22} + S_{23} \\ 1 \end{bmatrix} \tag{B.1}$$

We want to minimize the error

$$E = \sum_{\boldsymbol{u} \in R} |P(\mathcal{S}\boldsymbol{u}) - R(\boldsymbol{u})|^2,$$

with respect to $\mathcal{S}$. The error function for one pixel position $\boldsymbol{u}$ is then $e(\boldsymbol{u}) = P(\mathcal{S}\boldsymbol{u}) - R(\boldsymbol{u})$. The error function given to LM is the vector of $e(\boldsymbol{u})$ values produced by iterating $\boldsymbol{u}$ over all the discrete pixel positions in the reference patch. The parameters that LM modifies are the six elements $\mathcal{S}_{kl}$. We compute the elements of the Jacobian as

$$\frac{\partial e}{\partial \mathcal{S}_{kl}}(\boldsymbol{u}) = \frac{\partial P}{\partial x_1}\frac{\partial x_1}{\partial \mathcal{S}_{kl}} + \frac{\partial P}{\partial x_2}\frac{\partial x_2}{\partial \mathcal{S}_{kl}}.$$

Notice that the second term $R(\boldsymbol{u})$ in the function $e(\boldsymbol{u})$ drops out because it is constant w.r.t. $\mathcal{S}$. Also note that due to the form of the matrix multiplication in Eq. (B.1), only one of the two partial derivatives w.r.t. $\mathcal{S}_{kl}$ on the right is nonzero for any given subscript $kl$.

All that remains is to compute the partial derivatives $\partial P/\partial x_1$ and $\partial P/\partial x_2$ of $P$ w.r.t. to the components of $\boldsymbol{x}$. A low cost way to approximate these is to take the pixel values $p_{00}$, $p_{01}$, $p_{10}$ and $p_{11}$ from the four discrete locations closest to $\boldsymbol{x}$ in $P$ and compute the slope by interpolation. For example, if $d = x_2 - \lfloor x_2 \rfloor$, we have

$$\frac{\partial P}{\partial x_1} = (1-d)(p_{01} - p_{00}) + d(p_{11} - p_{10}).$$

The expression for $\partial P/\partial x_2$ is similar.

LM will of course find a local minimum of the error function rather than a global minimum. The initialization, based on affine adaptation, should in general be close enough to the correct value for the method to converge to a reasonable result. In practice, the results are quite good.

# Appendix C

# CD of Video Results

# References


[1] Brett Adams, Chitra Dorai, and Svetha Venkatesh. Study of shot length and motion as contributing factors to movie tempo. *ACM Multimedia*, 2000.

[2] Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, 2002.

[3] A. Aner and J. R. Kender. Video summaries through mosaic-based shot and scene clustering. In *European Conference on Computer Vision*, pages 388–402, Copenhagen, Denmark, 2002.

[4] Jo ao Paulo Costiera and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[5] N. Ayache and O. D. Faugeras. Hyper: a new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):44–54, January 1986.

[6] Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, September 2002.

[7] Adam Baumberg. Reliable feature matching across widely separated views. In *Conference on Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[8] Peter N. Belhumeur, João P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

[9] Serge Belongie, Jitendra Malik, and Jan Puzicha. Matching shapes. In *International Conference on Computer Vision*, pages 454–461, July 2001.

[10] Thomas O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, Miami, FL, December 1971.

[11] Stan Birchfield. Klt: An implementation of the kanade-lucas-tomasi feature tracker. `http://vision.stanford.edu/~birch/klt`, October 1998.

[12] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. In *IEEE Workshop on Visual Motion*, pages 179–186, 1991.

[13] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[14] R.A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence Journal*, 17(1-3):285–348, 1981.

[15] J. Brian Burns, Richard S. Weiss, and Edward M. Riseman. View variation of point-set and line-segment features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(1):51–68, January 1993.

[16] David Capel and Andrew Zisserman. Super-resolution from multiple views using learnt image models. In *Conference on Computer Vision and Pattern Recognition*, 2001.

[17] Owen Carmichael and Martial Hebert. Object recognition by a cascade of edge probes. In *British Machine Vision Conference*, 2002.

[18] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*, 1999.

[19] Hyun Sung Chang, Sanghoon Sull, and Sang Uk Lee. Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279, December 1999.

[20] Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):602–615, September 1998.

[21] Shih-Fu Chang, William Chen, and Hari Sundaram. Semantic visual templates: Linking visual features to semantics. In *International Conference on Image Processing*, 1998.

[22] Peter Cheeseman, Bob Kanefsky, Richard Kraft, and John Stutz. Super-resolved surface reconstruction from multiple images. Technical report, NASA Ames Research Center, December 1994.

[23] Carlo Colombo, Alberto Del Bimbo, and Pietro Pala. Semantics in visual information retrieval. *IEEE MultiMedia*, 6(3):38–53, 1999.

[24] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *International Conference on Computer Vision*, pages 1071–1076, Boston, MA, 1995.

[25] J. L. Crowley and A. C. Parker. A representation of shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:156–170, 1984.

[26] Serhan Dağtaş, Wasfi Al-Khatib, Arif Ghafoor, and Rangasami L. Kashyap. Models for motion-based video indexing and retrieval. *IEEE Transactions on Image Processing*, 9(1):88–101, January 2000.

[27] Glorianna Davenport, Thomas Aguierre Smith, and Natalio Pincever. Cinematic primitives for multimedia. *IEEE Computer Graphics & Applications*, 11(4):67–74, July 1991.

[28] P. Debevec, C. Taylor, and J. Malik. Modelling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[29] D. E. DiFranco, T.-J. Cham, and J. M. Rehg. Recovery of 3D articulated motion from 2D correspondences. Technical Report CRL 99/7, Compaq Cambridge Research Laboratory, 1999.

[30] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2001. Second edition.

[31] Roman Fablet and Patrick Bouthemy. Spatio-temporal segmentation and general motion characterization for video indexing and retrieval. In *10th DELOS Workshop on Audio-Visual Digital Libraries*, Santorini, Greece, June 1999.

[32] O. Faugeras, Q. T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[33] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986. 1986.

[34] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Conference on Computer Vision and Pattern Recognition*, volume II, pages 264–270, 2003.

[35] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *European Conference on Computer Vision*, 2004.

[36] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications ACM*, 24(6):381–395, June 1981.

[37] Andrew Fitzgibbon and Andrew Zisserman. Automatic 3d model acquisition and generation of new images from video sequences. In *European Signal Processing Conference*, Rhodes, Greece, 1998.

[38] Andrew Fitzgibbon and Andrew Zisserman. On affine invariant clustering and automatic cast listing in movies. In *European Conference on Computer Vision*, 2002.

[39] Andrew W. Fitzgibbon and Andrew Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *European Conference on Computer Vision*, pages 891–906. Springer-Verlag, June 2000.

[40] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, September 1995.

[41] D. Forsyth, J. Malik, M. Fleck, and J. Ponce. Primitives, perceptual organization and object recognition. Technical report, UC Berkeley, 1997.

[42] D.A. Forsyth and M.M. Fleck. Body plans. In *Conference on Computer Vision and Pattern Recognition*, pages 678–83, 1997.

[43] David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice-Hall, 2002.

[44] J. Gårding and T. Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. *International Journal of Computer Vision*, 17(2):163–191, 1996.

[45] C. W. Gear. Multibody grouping in moving objects. *International Journal of Computer Vision*, 29(2):133–150, 1998.

[46] M. Gelgon and P. Bouthemy. Determining a structured spatio-temporal representation of video content for efficient visualisation and indexing. Technical Report 1157, IRISA, December 1997.

[47] Bogdan Georgescu and Peter Meer. Point matching under large image deformations and illumination changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):674–689, June 2004.

[48] W. E. L. Grimson. The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence Journal*, 44(1-2):121–166, July 1990.

[49] W. E. L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

[50] Moving Picture Experts Group. Mpeg web site. `http://mpeg.telecomitalialab.com`.

[51] U. I. Gupta, D. T. Lee, and Y. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12:459–467, 1982.

[52] C. Harris and M. Stephens. A combined edge and corner detector. In $4^{\text{th}}$ *Alvey Vision Conference*, pages 189–192, Manchester, UK, 1988.

[53] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: global versus component-based approach. In *International Conference on Computer Vision*, pages 688–694, 2001.

[54] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. In *Conference on Computer Vision and Pattern Recognition*, pages 657–662, 2001.

[55] P. V. C. Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, CERN, 1959.

[56] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *International Conference on Computer Vision*, pages 102–111, 1987.

[57] T. Joshi, B. Vijayakumar, D. J. Kriegman, and J. Ponce. HOT curves for modeling and recognition of smooth curved 3D objects. *Image and Vision Computing*, 15(7):479–498, 1997.

[58] D. Keren, D. Cooper, and J. Subrahmonia. Describing complicated objects by implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):38–53, 1994.

[59] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[60] D. J. Kriegman and J. Ponce. On recognizing and positioning curved 3D objects from image contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1127–1137, December 1990.

[61] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based reconitiion scheme. In *International Conference on Computer Vision*, pages 238–249, 1988.

[62] Yehezkel Lamdan and Haim J. Wolfson. On the error analysis of 'geometric hashing'. In *Conference on Computer Vision and Pattern Recognition*, pages 22–27, Maui, Hawaii, 1991.

[63] Hyowon Lee, Alan F. Smeaton, Noel Murphy, Noel O'Conner, and Sean Marlow. User interface design for keyframe-based browsing of digital video. In *International Workshop on Image Analysis for Multimedia Interactive Services*, 2001.

[64] Zhibin Lei and Yun-Ting Lin. 3d shape inferencing and modeling for semantic video retrieval. In *Multimedia Storage and Archiving Systems, SPIE's Photonics East '96 Symposium*, Boston, MA, 1996.

[65] Rainer Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, August 2001.

[66] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[67] Tony Lindeberg and Jonas Gårding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In *European Conference on Computer Vision*, pages 389–400, Stockholm, Sweden, May 2-5 1994. Springer-Verlag Lecture Notes in Computer Science, vol. 800.

[68] J. Liu, J.L. Mundy, D. Forsyth, A. Zisserman, and C. Rothwell. Efficient recognition of rotationally symmetric surfaces and straight homogeneous generalized cylinders. In *Conference on Computer Vision and Pattern Recognition*, pages 123–128, New York City, NY, 1993.

[69] D. G. Lowe. The viewpoint consistency constraint. *International Journal of Computer Vision*, 1(1):57–72, 1987.

[70] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. In press.

[71] David G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

[72] David G. Lowe. Local feature view clustering for 3d object recognition. In *Conference on Computer Vision and Pattern Recognition*, volume I, pages 682–688, December 2001.

[73] Wei-Ying Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[74] Yu-Fei Ma and Hong-Jiang Zhang. Motion texture: A new motion based video representation. In *International Conference on Pattern Recognition*, pages 2:548–551, 2002.

[75] Shyjan Mahamud. *Discriminative Distance Measures for Object Detection*. PhD thesis, CMU, July 2002.

[76] Shyjan Mahamud and Martial Hebert. The optimal distance measure for object detection. In *Conference on Computer Vision and Pattern Recognition*, 2003.

[77] Shyjan Mahamud, Martial Hebert, and John Lafferty. Combining simple discriminators for object discrimination. In *European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[78] Shyjan Mahamud, Martial Hebert, Yasuhiro Omori, and Jean Ponce. Provably-convergent iterative methods for projective structure from motion. In *Conference on Computer Vision and Pattern Recognition*, pages 1018–1025, 2001.

[79] D. Marr. Representation and recognition of three-dimensional shapes. AI Memo 416, MIT, 1977.

[80] Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio Fujisawa. Efficient algorithms for finding maximum cliques of an overlap graph. Technical Report TR 1986-58, Institute for Systems Research, University of Maryland, 1986.

[81] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, volume I, pages 384–393, 2002.

[82] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. submitted to IJCV.

[83] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, pages 525–531, Vancouver, Canada, July 2001.

[84] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, volume I, pages 128–142, 2002.

[85] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *Conference on Computer Vision and Pattern Recognition*, 2003.

[86] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[87] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G.A. Watson, editor, *Numerical Analysis*, volume 1 of *Lecture Notes in Mathematics*. Springer-Verlag, 1977.

[88] P. Moreels, M. Maire, and P. Perona. Recognition by probabilistic hypothesis construction. In *European Conference on Computer Vision*, 2004.

[89] J. L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[90] J. L. Mundy, A. Zisserman, and D. Forsyth. *Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[91] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[92] V. S. Nalwa. Line-drawing interpretation: A mathematical framework. *International Journal of Computer Vision*, 2:103–124, 1988.

[93] M. R. Naphade and T. S. Huang. Semantic video indexing using a probabilistic framework. In *International Conference on Pattern Recognition*, pages 3:83–88, 2000.

[94] Ramakant Nevatia and Thomas O. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[95] Chong-Wah Ngo, Hong-Jiang Zhang, and Ting-Chuen Pong. Recent advances in content based video analysis. *International Journal of Image and Graphics*, 1(3):445–468, 2001.

[96] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, pages 555–562, 1998.

[97] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Conference on Computer Vision and Pattern Recognition*, Seattle, WA, 1994.

[98] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997.

[99] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest, and L. Van Gool. Structure and motion from image sequences. In Kahmen Grn, editor, *Proc. Conference on Optical 3-D Measurement Techniques V*, pages 251–258, Vienna, October 2001.

[100] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–26, August 1999.

[101] Marc Pollefeys, Frank Verbiest, and Luc Van Gool. Surviving dominant planes in uncalibrated structure and motion recovery. In A. Heyden et. al., editor, *European Conference on Computer Vision*, volume LNCS 2351, pages 837–851, 2002.

[102] J. Ponce, D. Chelberg, and W. Mann. Invariant properties of straight homogeneous generalized cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):951–966, September 1989.

[103] Jean Ponce. On computing metric upgrades of projective reconstructions under the rectangular pixel assumption. In *Second SMILE Workshop*, pages 18–27, 2000.

[104] Arthur R. Pope and David G. Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

[105] T. Poston and I. Stewart. *Catastrophy Theory and Its Applications*. Pitman, London, 1978.

[106] Philip Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *International Conference on Computer Vision*, pages 754–760, Bombay, India, 1998.

[107] Lawrence G. Roberts. *Machine Perception of Three-Dimensional Objects*. PhD thesis, Massachusetts Institute of Technology, May 1963.

[108] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision*, pages 700–714, Copenhagen, Denmark, 2002.

[109] Dan Roth. Learning to resolve natural language ambiguities: A unified approach. In *National Conference on Artifical Intelligence*, pages 806–813, 1998.

[110] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[111] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by backpropagating errors. *Nature*, 323(99):533–536, 1986.

[112] F. Schaffalitzky and A. Zisserman. Automated scene matching in movies. In *Proceedings of the Challenge of Image and Video Retrieval*, London, 2002.

[113] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *European Conference on Computer Vision*, volume I, pages 414–431, 2002.

[114] R. E. Schapire. The boosting approach to machine learning, an overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[115] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), May 1997.

[116] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[117] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Conference on Computer Vision and Pattern Recognition*, 2000.

[118] A. Selinger and R.C. Nelson. A perceptual grouping hierarchy for appearance-based 3D object recognition. *Computer Vision and Image Understanding*, 76(1):83–92, 1999.

[119] A. Sethi, D. Renaudie, D.J. Kriegman, and J. Ponce. Curve and surface duals and the recognition of curved 3D objects from their silhouette. In *International Journal of Computer Vision*, 2002. Submitted.

[120] Jianbo Shi and Carlo Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, 1994.

[121] K. Siddiqi and B. B. Kimia. Parts of visual form: computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, March 1995.

[122] Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. Object level grouping for video shots. In *European Conference on Computer Vision*, 2004.

[123] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[124] J. V. Stone. Shape from texture: textural invariance and the problem of scale in perspective images of surfaces. In *British Machine Vision Conference*, pages 181–186, 1990.

[125] Steve Sullivan and Jean Ponce. Automatic model construction, pose estimation, and object recognition from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1091–1096, October 1998.

[126] Yap-Peng Tan, Sanjeev R. Kulkarni, and Peter J. Ramadge. A framework for measuring video similarity and its application to video query by example. In *IEEE International Conference on Image Processing*, 1999.

[127] Dennis Tell and Stefan Carlsson. Wide baseline point matching using affine invariants computed from intensity profiles. In *Proc 6th ECCV*, pages 814–828, Dublin, Ireland, June 2000. Springer LNCS 1842-1843.

[128] D. Thompson and J. Mundy. Three-dimensional model matching from an unconstrained viewpoint. In *International Conference on Robotics and Automation*, pages 208–220, Raleigh, NC, 1987.

[129] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[130] P. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[131] Philip Hilaire Sean Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, University of Oxford, 1995.

[132] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms*, pages 298–372, Corfu, Greece, September 1999. Spinger-Verlag. LNCS 1883.

[133] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Winter 1991.

[134] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *Conference on Computer Vision and Pattern Recognition*, Maui, Hawaii, 1991.

[135] Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affinely invariant neighbourhoods. *International Journal of Computer Vision*, 2004. In press.

[136] Tinne Tuytelaars and Luc J. Van Gool. Content-based image retrieval based on local affinely invariant regions. In *Visual Information and Information Systems*, pages 493–500, 1999.

[137] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, 1995.

[138] VIBES. Video browsing, exploration and structuring. `http://www.inrialpes.fr/movi/people/Triggs/vibes`.

[139] René Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). In *Conference on Computer Vision and Pattern Recognition*, 2003.

[140] B. Vijayakumar, D.J. Kriegman, and J. Ponce. Invariant-based recognition of complex curved 3d objects from image contours. *Computer Vision and Image Understanding*, 72(3):287–303, 1998.

[141] Paul Viola and Michael Jones. Robust real-time object detection. Technical Report 2001/01, Compaq Cambridge Research Laboratory, February 2001.

[142] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *International Conference on Computer Vision*, pages 250–258, 87.

[143] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, 2000.

[144] Daphna Weinshall and Carlo Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):512–517, 1995.

[145] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. In L.C. Jain et al., editor, *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, chapter 11, pages 355–396. CRC Press, 1999.

[146] Yi Wu, Yueting Zhuang, and Yunhe Pan. Content-based video similarity model. *ACM Multimedia*, 2000.

[147] Mourad Zerroug and Ramakant Nevatia. From an intensity image to 3D segmented descriptions. In J. Ponce, A. Zisserman, and M. Hebert, editors, *Object Representation in Computer Vision II*, number 1144 in LNCS, pages 11–24. Springer-Verlag, 1996.

[148] Li Zhao, Wei Qi, Stan Z. Li, Shi-Qiang Yang, and H. J. Zhang. Key-frame extraction and shot retrieval using nearest feature line (nfl). In *International Workshop on Multimedia Information Retrieval*, 2000.

[149] Di Zhong and Shih-Fu Chang. An integrated approach for content-based video object segmentation and retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1259–1268, December 1999.

[150] S. C. Zhu and A. L. Yuille. FORMS: a flexible object recognition and modeling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.

[151] Andrew Zisserman, Andrew Fitzgibbon, and Geoff Cross. Vhs to vrml: 3d graphical models from video sequences. In *IEEE International Conference on Multimedia and Systems*, Florence, 1999.

# Author's Biography

Fredrick Henry Rothganger was in born Hong Kong on December 21, 1968. He lived in Vietnam from 1972 to 1975, and then in the Philippines from 1976 until 1987. He graduated *cum laud* from Central Bible College in 1990 with a degree in pastoral ministries. He worked in a Christian short-term volunteer organization as a computer technician until moving to Boston, Massachusetts, in 1994 to pursue graduate studies in computer science. He graduated from the University of Massachusetts at Boston in 1997 with a masters degree. In 1999 he moved to Urbana, Illinois, to pursue doctoral studies in computer science. Following completion of his Ph.D., Fredrick will move to Albuquerque, New Mexico, to begin work as a senior member of technical staff at Sandia National Laboratories.